KODENESIA
71 SOAL ALGORITMA DASAR

Daftar Isi

1 Kata Pengantar

Terimakasih. Anda telah membaca buku ini. Setidaknya sampai pada paragraf pertama kata pengantar. Buku yang isinya soal tanpa jawaban ini lahir karena sebuah alasan. Ada alasan utama, serta sekian butir penyebab lainnya. Ketrampilan menulis kode program dengan bahasa tertentu, memerlukan sekian jam latihan. Itu adalah sebuah keniscayaan. Hanya kejadian ajaib yang bisa menghadirkan seseorang yang mahir tanpa melalui fase latihan. Mereka yang belajar pemrograman memerlukan soal untuk bahan latihan. Buku ini menyediakannya.

Untuk siapakah buku ini ditujukan? Siapa saja. Tak ada batasan. Mereka yang sedang belajar menulis kode program adalah sasaran utamanya. Digunakan sebagai bahan latihan mengasah nalar algoritma dasar. Bisa untuk pemula, atau pemrogram berpengalaman yang mempelajari bahasa baru. Membuat program untuk menjawab soal bisa membantu dalam menguasai sebuah bahasa. Pembaca lain yang tidak terkait dengan pemrograman bisa menggunakan soal untuk latihan pemecahan masalah. Memang sebagian soal sangat terkait dengan teknologi informasi, tetapi cukup banyak yang sifatnya umum.

Dosen dapat menggunakan buku ini sebagai sumber inspirasi ketika membuat soal ujian. Penulis soal (misalnya untuk proses seleksi karyawan sebagai *programmer*) juga bisa memanfaatkan dengan cara yang sama. Tak ada larangan untuk menggunakan soal dari buku ini. Memberi kredit untuk penulis buku ketika mengutip isinya adalah baik, tapi bukan sebuah keharusan ketika mengutip dari buku ini.

Tujuhpuluhsatu soal bisa banyak, bisa sedikit. Saya berusaha membuatnya tidak sama. Bisa jadi satu soal dengan lainnya ada yang mirip. Mungkin ada beberapa soal yang bisa diselesaikan dengan cara yang identik. Tak tertutup kemungkinan ada pernyataan yang tidak benar. Nah, yang ini harusnya tidak terjadi. Jika ada, adalah koreksi dari pembaca yang bisa membantu saya mengetahui letak salahnya.

Banyak sahabat yang terlibat dalam penyusunan buku ini. Sebagian besar memberi kontribusi sebagai teman diskusi. Terimakasih saya untuk semuanya. Gibransyah Fakhri membantu melahirkan beberapa ide, ketika sebagian soal dituangkan dalam media *online*. Kaka Prakasa, Utian Ayuba, dan Ade Muhammad adalah teman yang sering mencetuskan ide segar saat berkomunikasi di ruang maya #kpli-bogor. Arif Syamsudin banyak menjelaskan kepada saya pernak-pernik terkait dunia perbukuan. Untuk Yanti Rusmawati yang menyebut dirinya petualang, terima kasih atas kebaikannya menulis kata pengantar. Beliau adalah sedikit orang yang punya pengetahuan tentang hal yang rumit dalam ranah komputasi digital. Ada banyak rekan lain yang tak bisa saya tulis satu persatu namanya di sini, kepada mereka ucapan terima kasih yang tulus dari saya.

Buku ini ditulis menggunakan penyunting teks Vim. Dokumen disusun dalam for-

mat LaTeX yang diproses menjadi PDF menggunakan paket Texlive. Semuanya berjalan pada Slackware. Lingkungan perangkat lunak bebas ini menyediakan alat yang lengkap untuk menulis buku, bahkan sampai hasil akhir yang akan dikirim ke percetakan. Saya berterimakasih pada mereka yang menghasilkan program komputer seperti ini.

Untuk istri tercinta yang membahagiakan dalam keluarga, mungkin tak sepenuhnya mengerti perihal apakah gerangan buku ini. Untuk Karim dan Ali yang mungkin menyukai isinya di kemudian hari. Kepada Tuhan kita bersyukur atas segala nikmat.

Yanmarshus Bachtiar

2 Sekapur Sirih

Ditulis oleh *Yanti Rusmawati*[1]

> Kita manusia ini semua melihat satu bulan saja, tetapi banyak jalan yang dapat kita tempuh untuk sampai ke puncak yang terdekat dengannya. Kadang-kadang kalau kita tersesat, kita memutuskan untuk mencoba jalan orang lain, tapi tujuan akhirnya menemukan penyempurnaan hidup. (*Musashi*:683, Eiji Yoshikawa, 2001)

Manusia sebagai makhluk berakal selalu mencoba berkomunikasi dengan berbagai makhluk lainnya, baik yang bernyawa maupun tidak. Tak peduli makhluk itu ada di dunia nyata, dunia maya, dunia lain, bahkan dunia antar galaksi. Dalam setiap komunikasi, diperlukan bahasa yang bisa dipahami oleh kedua belah pihak yang terlibat. Sejarah telah membuktikan pula bahwa manusia selalu bisa menemukan cara untuk berkomunikasi, baik dengan bahasa verbal, isyarat, maupun tulisan.

Salah satu komunikasi yang cukup sering dibangun adalah komunikasi dengan mesin yang bernama komputer. Melalui sebuah program - yang mewakili persoalan yang ingin kita pecahkan - kita berkomunikasi dengan komputer. Berbagai bahasa pemrograman muncul seiring perkembangan teknologi komputer dan informasi. Tak heran kita lihat begitu berlimpahnya buku-buku yang memaparkan bahasa pemrograman A, B, sampai dengan Z, sehingga terkadang membingungkan mana yang perlu dipilih dan dipelajari.

Lantas apa yang unik dari buku *Kodenesia* ini? Fokus pembaca tidak lagi melulu pada satu jenis bahasa pemrograman. Bertolak dari soal-soal, penulis berusaha menularkan logika sistematis yang semestinya ada dalam kode program yang kita buat. Buku ini bukan hanya mengenalkan kita pada berbagai soal menarik dan menggelitik kalbu, namun juga membawa kita memahami persoalan dengan berpikir aktif. Kemampuan penulis memaparkan hal yang kompleks dan 'seram' menjadi sesuatu yang seindah karya seni sudah selayaknya mendapat acungan jempol.

Tujuh puluh satu soal memang bisa dibilang banyak, bisa juga sedikit, tergantung dari sisi mana kita melihat. Bagi saya pribadi, satu soal saja dapat memiliki banyak alternatif pemecahan masalah, hingga mungkin sulit untuk mengatakan: "Saya sudah tamat membaca buku *Kodenesia*-nya pak Yan". Semoga ini baru buku seri perdana, karena kita butuh buku-buku yang mencerahkan lainnya dari penulis (mungkin tentang *filosofi hacker*? :)

[1] Petualang logika di Bandung (1989-1994), Depok(1999-2001), dan Manchester UK (2009 - *hopefully* 2014). Fans berat Miyamoto Musashi dan Kungfu Panda.

3 Pendahuluan

Soal dalam buku ini disajikan dalam kerangka yang longgar. Tidak melibatkan pernyataan matematika yang formal. Ada sedikit soal yang menyertakan kalimat matematis dalam bentuk sederhana. Deskripsi permasalahan dibuat sejelas mungkin. Beberapa soal menggunakan kalimat ringkas dan jelas. Sebagian lagi berbentuk cerita, bahkan mirip dongeng.

Program dibuat untuk menjawab soal. Satu program untuk satu soal. Satu program untuk menjawab semuanya tentu saja bisa. Tapi bukan itu yang hendak dituju. Program membaca data dari sebuah *file*. Melakukan proses terhadap data, kemudian menghasilkan *output*. Misalnya perintah dalam soal kita adalah : kalikan dua buah bilangan A dan B. Program membaca data A dan B dari sebuah *file*, melakukan proses A x B, lalu mengeluarkan hasilnya. Cara ini yang dianjurkan untuk menjawab soal. Cara lain juga boleh dilakukan, sejauh maksud dari program adalah menjawab kasus yang diuraikan dalam soal.

File yang dibaca sebagai sumber data adalah *file* teks. Data disusun dalam aturan tertentu. Di setiap akhir sebuah soal ada penjelasan dan contoh data yang akan diproses. Secara umum, apabila data terdiri dari sejumlah angka, maka angka-angka ini dipisahkan oleh satu buah spasi. Demikian juga untuk data yang terdiri dari beberapa kata, pemisahnya sama, yaitu satu buah spasi.

Jumlah baris dalam *file* ini bisa lebih dari satu. Setiap baris diproses, kemudian dihasilkan *output* sesuai dengan data pada baris tersebut. Contohnya sebagai berikut.

```
20 5
4 3
11 11
1200 1
6 6
```

Setiap baris ada dua angka yang dipisahkan oleh satu spasi. Data ini akan diproses oleh sebuah program yang mengerjakan perkalian dua buah bilangan. Ada 5 baris data. Program membaca setiap baris, mengalikan kedua bilangan, kemudian menampilkan *output*. Hasilnya adalah seperti berikut.

```
100
12
121
1200
36
```

Google Codejam dan Facebook Hackercup adalah kompetisi pemrograman yang diadakan setiap tahun secara *online*. Keduanya mempunyai model yang mirip dalam cara menjawab soal. Begitu juga untuk buku ini. Kita meminjam cara yang sama. Baca *input*, proses, hasilkan *output*.

Di akhir setiap soal, disertakan contoh untuk data *input* dan *output*. Bentuknya dalam dua kolom. Kolom kiri adalah *input*, kolom kanan adalah *output*. Jumlah data untuk setiap contoh adalah 3 baris. Adakalanya contoh tidak bisa dibuat lengkap karena keterbatasan ruang. Kembali ke soal perkalian 2 angka tadi, contoh *input* dan *output* adalah seperti berikut.

```
20 5                                    h#1: 100
4 3                                     h#2: 12
11 11                                   h#3: 121
```

Hasil dari program sebaiknya disimpan dalam sebuah *file*. Dengan adanya dua *file* yakni *input* dan *output*, dapat dimanfaatkan untuk menguji kebenaran hasil program. Tentu saja dilakukan oleh program lain yang sudah berfungsi dengan benar.

Soal yang akan dikerjakan bisa dipilih secara acak. Tidak ada urutan dalam tingkat kesulitan. Soal yang di awal belum tentu lebih mudah dari soal yang di akhir. Perihal derajat kesulitan, soal dalam buku ini berada dalam tingkat yang mudah. Sebagian kecil ada yang di atasnya. Sedangkan yang rumit, saya yakin tidak ditemukan diantara 71 soal ini.

Mengerjakan soal dalam buku ini akan melatih beberapa pengetahuan kita dalam sebuah bahasa. Apa saja? Membaca data dari *file*, penggunaan konstruksi dasar seperti percabangan, perulangan (*loop*), struktur data teks, angka, *array*, penggunaan *stack*, operasi matematika dasar, operasi logika, dan lain-lain.

4 Soal

Ada sebuah soal sederhana yang disebut dengan FizzBuzz. Soal ini berhubungan dengan salah satu operasi matematika, yaitu pembagian. Atau bisa juga perkalian. Hasil yang diinginkan dari program berupa angka, kata Fizz, dan kata Buzz. Pada proses seleksi penerimaan karyawan untuk menempati posisi *programmer*, ada yang menjadikan soal ini sebagai sebuah ukuran seseorang mempunyai kemampuan dasar algoritma yang baik atau tidak.[2] Berikut ini adalah deskripsi dari soal FizzBuzz.

Buatlah sebuah program yang menghasilkan 100 baris angka, berurutan dari 1 sampai dengan 100. Apabila sebuah angka habis dibagi 3, maka tampilkan kata Fizz di sebelahnya. Jika angkanya habis dibagi 5, tampilkan kata Buzz di sebelahnya. Bila angka tersebut habis dibagi 3 dan habis dibagi 5, tampilkan kata FizzBuzz di sebelah angka tersebut.

Di sebelah ini adalah contoh hasil program sebagai jawaban soal FizzBuzz. Hasil yang ditampilkan hanya baris 1 sampai baris 17.

Untuk menyelesaikan soal ini setidaknya kita memahami proses perulangan, percabangan, dan operator modulo. Proses perulangan (*loop*) untuk menghasilkan angka 1 sampai 100. Alur percabangan bersama dengan operator modulo digunakan untuk menguji apakah sebuah angka habis dibagi oleh bilangan tertentu.

Membuat program untuk menyelesaikan soal FizzBuzz ini layak dikerjakan sebagai sebuah pemanasan untuk soal-soal selanjutnya. Apabila Anda berhasil mengerjakannya dalam waktu kurang dari 15 menit, maka bergembiralah. Permulaan yang layak dirayakan.

```
1
2
3 Fizz
4
5 Buzz
6 Fizz
7
8
9 Fizz
10 Buzz
11
12 Fizz
13
14
15 FizzBuzz
16
17
```

[2] A surprisingly large fraction of applicants, even those with masters degrees and PhDs in computer science, fail during interviews when asked to carry out basic programming tasks. For example, I've personally interviewed graduates who can't answer "Write a loop that counts from 1 to 10" or "What's the number after F in hexadecimal?" Less trivially, I've interviewed many candidates who can't use recursion to solve a real problem. These are basic skills; anyone who lacks them probably hasn't done much programming. http://www.kegel.com/academy/getting-hired.html

4.1 Menutup yang Terbuka

Dalam bahasa pemrograman lazim dijumpai penggunaan tanda kurung. Tanda kurung yang digunakan berupa karakter `(` atau `{` atau karakter `[`. Pemakaian tanda kurung ini umumnya secara berpasangan. Ada tanda kurung pembuka dan ada tanda kurung penutup.

Dalam sebuah teks, tanda kurung dianggap seimbang kalau digunakan secara berpasangan. Ada tanda kurung pembuka, harus ada tanda kurung penutup. Jika pembukanya tanda kurung `(`, maka penutupnya adalah `)`. Demikian juga untuk tanda kurung `{` mempunyai pasangan `}`. Dan tanda kurung `[` berpasangan dengan `]`.

Kita akan menentukan apakah dalam sebuah teks tanda kurungnya seimbang atau tidak. Seperti penjelasan di atas, seimbang artinya tanda kurung pembuka harus disertai dengan penutup. Tanda kurung dalam teks ini bisa banyak, dan sebuah tanda kurung bisa berada dalam tanda kurung lainnya. Penggunaan tanda kurung yang ada dalam tanda kurung lainnya harus dalam urutan yang benar. Berikut ini adalah contoh penggunaan tanda kurung yang seimbang.

```
makna(ada)dalam{kata}
seperti((hati)yang[dibuai]mimpi)
mawar{tidak{selal[u]}mer(ah)}
```

Contoh berikut ini untuk penggunaan tanda kurung yang tidak seimbang

```
dalam(persimpangan
meniti{hu[tan}]cemara
merpati{{tak{ingkar}]}janji}
```

Data soal. Setiap baris berisi teks yang terdiri dari karakter a-z serta kombinasi tanda kurung. Panjang satu baris maksimal 1000 karakter, minimal 1 karakter. Hasil dari program adalah kata YA atau TIDAK. Apabila tanda kurung seimbang, tampilkan kata YA. Jika tidak seimbang, tampilkan kata TIDAK.

```
hkd(ffg{jknn}sdfhngjtttg)          h#1: YA
h(((sss[](c))))xyz                 h#2: YA
hhiwwdf[dfgg)bqwci}dff             h#3: TIDAK
```

4.2 Urutan Terpanjang

Ada sebuah kumpulan angka. Dalam himpunan tersebut setiap angka adalah unik. Tidak ada angka yang sama antara satu dengan lainnya. Susunan angka tersebut tidak berurutan. Berikut ini contoh himpunan angka seperti deskripsi di atas.

```
5 8 6 2 44 45 22 46 1 7 9 21 24 23 3 25 26 27 28
```

Kita akan mencari angka berdampingan yang paling panjang. Caranya, dari angka-angka ini bisa kita susun kembali menjadi beberapa kelompok. Setiap kelompok harus terdiri dari angka yang berurutan. Hasil penyusunan ulang menjadi beberapa kelompok seperti berikut ini.

```
1 2 3
5 6 7 8 9
21 22 23 24 25 26 27 28
44 45 46
```

Dari hasil penyusunan ulang ini, kelompok yang paling panjang adalah kelompok ke-3, yaitu sebanyak delapan angka. Berurut dari 21 sampai dengan 28. Jawaban kita untuk contoh ini adalah 8.

Data soal. Setiap baris terdiri dari 1 angka atau lebih. Paling banyak adalah 1000 angka. Setiap angka dipisahkan oleh satu spasi. Angka berada dalam rentang 1 sampai 10^7. Program mencari urutan terpanjang angka yang berdampingan.

```
4 8 9 7 1 12 13 99 43 44            h#1: 3
1 2 3 4 5 6 7 8 9 22 23             h#2: 9
1 2 8 9 11 12 22 23 67 68 89 90     h#3: 2
```

4.3 Percakapan Rahasia

Suatu masa ketika masih kanak-kanak, orang muda di lingkungan rumah saya mempunyai cara untuk merahasiakan percakapan mereka. Kata yang mereka ucapkan terdengar asing dan tidak bisa dipahami. Sekian tahun kemudian saya baru mengerti, ternyata caranya tidaklah rumit. Setiap kata dibalik mengucapkannya. Misalnya untuk kata BUKU diucapkan UKUB. Dalam bentuk kalimat contohnya sebagai berikut.

```
Kalimat normal  : KITA AKAN PERGI
Kalimat rahasia : ATIK NAKA IGREP
```

Data soal. Setiap baris berisi kalimat dalam bentuk pengucapan rahasia. Program dibuat untuk memecahkan kalimat rahasia menjadi kalimat yang dapat dibaca dengan normal. Setiap kata terdiri dari huruf a-z. Pemisah antar kata adalah satu buah spasi. Satu kalimat maksimal terdiri dari 1000 karakter termasuk spasi. Batas minimal satu kalimat adalah satu kata yang sekurang-kurangnya terdiri dari 3 huruf.

```
italem irad irigayaj            h#1: melati dari jayagiri
iadab itsap ulalreb             h#2: badai pasti berlalu
nalub kusutret gnalali          h#3: bulan tertusuk ilalang
```

4.4 Kata yang Kaya

Sebuah kata terdiri dari sejumlah huruf. Misalnya kata JUJUR, terdiri dari 5 huruf. Dalam kata JUJUR tersebut ada huruf yang digunakan lebih dari satu kali, yaitu huruf J dan huruf U, masing-masing digunakan 2 kali. Jika kita hanya menghitung jumlah huruf yang unik, maka kata JUJUR hanya menggunakan 3 huruf yaitu J, U, dan R.

Dalam contoh kalimat berikut ini, kita akan menghitung jumlah huruf unik dalam setiap kata.

```
bulan bersembunyi ketika matahari menghampiri pagi
5     9           5      6        9           4
```

Kata yang menggunakan huruf unik paling banyak adalah kata bersembunyi dan menghampiri, yaitu 9 huruf.

Data soal. Setiap baris berisi sejumlah kata. Pemisah antar kata adalah satu buah spasi. Kata terdiri dari huruf a-z. Maksimal jumlah karakter dalam satu baris adalah 1000. Hasil dari program adalah jumlah huruf unik terbanyak yang dapat ditemukan diantara kata-kata tersebut.

```
kata di ujung pena                  h#1: 4
juli di bulan juni                  h#2: 5
daun di atas bantal                 h#3: 5
```

4.5 Hilang Dalam Keramaian

Menemukan jarum dalam tumpukan jerami. Sebuah pepatah lama. Kiasan betapa sulitnya pekerjaan seperti itu untuk dilakukan. Kasus ini merupakan soal yang tak jarang terjadi dalam pemrograman. Ada yang mudah diselesaikan, pasti banyak juga yang merepotkan untuk dipecahkan. Mari kita lihat kasus yang sama dalam soal ini.

Sebuah kumpulan angka disusun secara berurutan dari yang terkecil sampai yang terbesar. Jika ada satu angka yang hilang dalam urutan tersebut, cukup mudah untuk menemukannya. Bila angkanya tidak terlalu banyak, misalnya 10 angka, tanpa bantuan alat, manusia bisa mencarinya. Misalnya pada urutan angka berikut ini.

```
14   15   16   18   19   20
```

Dapat dilihat dengan mudah angka yang hilang adalah angka 17. Sekarang bagaimana jika spasi dalam susunan angka tersebut kita hilangkan? Susunan angka akan menjadi seperti berikut ini.

```
141516181920
```

Dengan hanya memberikan informasi bahwa angka tersebut berurutan, dan ada satu angka yang hilang dalam urutan tersebut, menemukan angka yang hilang tidak lagi semudah sebelumnya. Ada usaha ekstra untuk melihat terlebih dahulu seperti apakah urutan angka itu sesungguhnya.

Data soal. Setiap baris berisi angka berurutan dari kecil ke besar. Angka minimal adalah 1, dan angka maksimal adalah 10^6. Tidak ada pemisah antara angka dalam urutan tersebut. Panjang minimal satu baris soal adalah 3 angka, maksimal 1000 angka. Angka yang hilang berada di tengah, bukan di awal atau di akhir. Program dibuat untuk menemukan angka yang hilang.

```
23242526272830                          h#1: 29
10110210310410510610710810911112        h#2: 110
12346789                                h#3: 5
```

4.6 Tetangga Sama Warna

Sebuah gambar digital terdiri dari susunan *pixel* dengan lebar dan tinggi tertentu. Gambar dalam format *greyscale*. Ada 16 derajat keabu-abuan dengan skala dari 0 sampai 15. Warna hitam bernilai 0 dan warna putih bernilai 15.

Dalam sebuah perangkat lunak pengolah gambar, biasanya terdapat fitur untuk memilih warna yang ada pada gambar. Lakukan seleksi untuk warna hitam, maka semua *pixel* yang berwarna hitam dalam gambar itu akan terpilih. Ada juga variasi lain. Penunjuk *mouse* diarahkan pada sebuah *pixel*, lalu klik. Kita sebut *pixel* yang diklik ini dengan *pixel* T. *Pixel* yang berdampingan ikut terpilih jika warnanya sama dengan *pixel* T. Selanjutnya, *pixel* lain yang berdampingan dengan *pixel* yang terpilih dan memiliki warna sama, juga akan ikut terpilih. Proses ini berlanjut sampai tidak ada lagi *pixel* yang berdampingan yang memiliki warna sama.

Berikut ini sebuah gambar 8 x 8 *pixel* disajikan dalam nilai warna setiap pixel.

```
0  0  0  0  0  1  1  1
1  1  2  8* 2  2  2  3
7  6  8* 8* 8* 5  5  8
4  4 +8* 8* 9  0  0  0
3  3  3  3  8* 9  1  1
1  1  1  8* 8* 4  8  8
8  8  2  8* 2  2  0  0
3  4  2  2  0  0  0  0
```

Jika T adalah *pixel* yang terpilih, ada 8 *pixel* yang berdampingan dengan T seperti diagram berikut:

```
1  2  3
4  T  5
6  7  8
```

Pixel yang berada pada kolom ke-3 baris ke-4 diklik, berapa banyak *pixel* yang akan terpilih? *Pixel* yang diklik diberi tanda plus (+) di samping kiri angka. Mari kita lihat warna *pixel* tersebut. Nilai warnanya adalah 8. Maka tetangga *pixel* tersebut apabila bernilai 8 akan ikut terpilih. Seterusnya, jika ada *pixel* bernilai 8 yang berdampingan dengan *pixel* yang ikut terpilih, maka *pixel* tersebut juga ikut terpilih. Dari contoh ini, jumlah *pixel* yang terpilih adalah 10. *Pixel* yang terpilih diberi tanda bintang (*).

Posisi *pixel* disajikan dalam `(kolom, baris)`. Kolom ke-1 dimulai dari kiri. Sedangkan baris ke-1 dimulai dari atas. *Pixel* `1,1` adalah *pixel* yang berada di sudut kiri atas.

Data soal. Setiap baris berisi data *pixel* sebuah gambar. Data berupa angka yang mewakili nilai warna sebuah *pixel*. Setiap angka dipisahkan oleh satu spasi. Satu gambar mempunyai sisi yang sama panjang. Minimal terdiri dari 4 *pixel*, dan maksimal 10^4 *pixel*. Rentang nilai warna *pixel* mulai dari 0 sampai dengan 15. Dua angka pertama adalah koordinat *pixel* yang diklik. Angka selanjutnya adalah data setiap *pixel* yang ada dalam gambar. Misalnya ada 9 data *pixel*. Artinya gambar berukuran 3 x 3 *pixel*. Data ini perlu disesuaikan ketika mengolahnya. Tiga data *pixel* pertama akan berada di baris pertama, 3 data berikutnya pada baris ke-2, dan 3 data lagi untuk baris ke-3. Program menghitung berapa banyak *pixel* yang akan terpilih.

```
2 2 0 0 5 5                         h#1: 2
1 1 6 6 6 6 6 6 6 6 6               h#2: 9
1 3 9 8 7 7 7 7 4 7 7               h#3: 1
```

4.7 Penjumlahan yang Dibatalkan

Dua orang bermain penjumlahan. Namanya Kaka dan Dede. Kaka menyebutkan sebuah angka, Dede menyebutkan total penjumlahannya. Misalnya Kaka mulai dengan angka 5. Dede menjawab 5. Dilanjutkan Kaka menyebut 10. Dede menjawab 15. Kenapa 15? Karena total sebelumnya adalah 5. Jadi total setelah ditambah 10 adalah 15. Selanjutnya Kaka menyebut angka 3. Dede menjawab 18. Demikian permainan ini seterusnya.

Selain menyebutkan sebuah angka, Kaka bisa juga menyebut kata BATAL. Jika Kaka menyebut kata BATAL, maka Dede akan membatalkan penjumlahan terakhir, dan menyebutkan nilai total yang sebelumnya.

Kembali kita lihat di paragraf pertama cerita ini. Dede terakhir menjawab 18. Kita lanjutkan, Kaka menyebut angka 2. Dede menjawab 20. Kaka menyebut BATAL. Sekarang Dede membatalkan penjumlahan terakhir, yaitu 18 + 2, dan Dede menyebut nilai total sebelumnya, yaitu 18.

Permainan berlanjut, Kaka menyebut 7. Dede menjawab 25. Angka 25 adalah hasil penjumlahan dari angka 18 (total terakhir) dengan angka 7. Demikian permainan ini seterusnya. Kaka menyebut sebuah angka, atau kata BATAL. Dede menjawab sesuai aturan permainan ini. Dengan menuliskan urutan angka atau kata BATAL yang disebutkan oleh Kaka, kita dapat mengetahui jawaban terakhir yang diberikan oleh Dede. Contohnya seperti berikut ini. Kata BATAL ditulis dengan satu huruf B.

```
2  10  4  B  10  20  8  5  5  B  4  2
```

Jawaban terakhir Dede adalah 61.

Data soal. Setiap baris terdiri dari angka dan huruf B yang dipisahkan oleh satu buah spasi. Tidak ada huruf B yang berurutan. Angka minimal adalah 0, maksimal 1000. Jumlah angka dan huruf B dalam satu baris tidak lebih dari 1000. Hasil dari program adalah angka terakhir sesuai deskripsi soal ini.

```
2 2 B 10 B 7 B 5 B 11 B              h#1: 2
1 1 1 1 1 1 1 1 1 1 1                h#2: 11
10 B 2 2 B 2 2                       h#3: 6
```

4.8 Berhitung dengan Jari

Mari berhitung dengan jari. Gunakan sebelah tangan saja. Mulai dari jempol, kemudian telunjuk, jari tengah, jari manis, dan terakhir kelingking. Kita mulai dari angka 1 di jari jempol. Angka 2 pada jari telunjuk, dan seterusnya sampai angka 5 di kelingking. Apabila kita akan melanjutkan ke angka 6, maka kembali ke jari manis, angka 7 di jari tengah, angka 8 di jari telunjuk, dan 9 di jari jempol. Jika diteruskan lagi, maka angka 10 di jari telunjuk, dan seterusnya bolak-balik pada jari yang lima tersebut. Misalnya kita berhitung dari 1 sampai 7, maka angka terakhir, yaitu angka 7, jatuh di jari tengah.

Data soal. Setiap baris berisi dua angka, yaitu angka awal dan angka akhir. Angka dipisahkan dengan satu spasi. Angka pertama dimulai dari jari jempol. Teruskan sampai angka terakhir seperti cara di atas. Hasil dari program adalah nama jari tempat angka terakhir berada. Nama jari adalah : jempol, telunjuk, tengah, manis, kelingking.

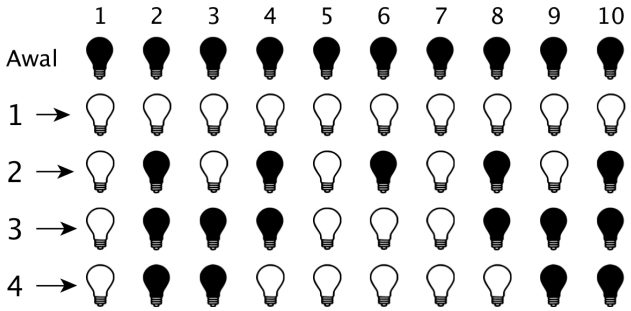
```
1 10                                    h#1: telunjuk
102 104                                 h#2: tengah
3 11                                    h#3: jempol
```

4.9 Lampu yang Hidup Mati

Dalam sebuah ruangan terdapat sejumlah lampu yang dipasang di langit-langit. Hal yang menarik dari instalasi lampu ini adalah hanya memiliki satu buah tombol. Tombol ini jika ditekan akan menyebabkan lampu yang sedang menyala akan mati, dan lampu yang sedang mati akan menyala. Setiap lampu diberi nomor urut dari 1 sampai dengan n. Dimana n adalah jumlah lampu.

Jika kondisi awal semua lampu dalam keadaan mati, maka pertama kali tombol ditekan, semua lampu dengan nomor kelipatan 1, ini artinya semua lampu, akan menyala. Jika tombol tersebut ditekan untuk kedua kali, maka lampu yang bernomor kelipatan 2, jika menyala akan menjadi padam, dan sebaliknya jika padam jadi menyala. Ditekan lagi tombol ini untuk yang ketiga kali, maka semua lampu yang bernomor kelipatan 3, jika mati akan menyala, dan jika menyala akan mati. Demikian seterusnya jika tombol tersebut ditekan.

Misalnya ada 10 lampu bernomor 1 sampai 10. Awalnya semua lampu dalam keadaan mati. Setelah 4 kali saklar ditekan, berapa buah lampu yang menyala? Dalam gambar berikut ini, lampu berwarna hitam adalah lampu dalam keadaan mati. Lampu yang berwarna putih dalam keadaan menyala.



Setelah saklar ditekan 4 kali, jumlah lampu yang menyala adalah 6 lampu yaitu lampu 1, 4, 5, 6, 7, dan 8.

Data soal. Dalam soal ini akan dicari jumlah lampu yang menyala setelah tombol ditekan sebanyak n kali. Kondisi awal untuk setiap baris data adalah sama, yaitu semua lampu mati. Setiap baris berisi 2 angka yang dipisahkan oleh satu spasi. Angka pertama adalah jumlah lampu. Angka kedua adalah berapa kali saklar ditekan. Jumlah lampu minimal adalah 1, maksimal 1000. Saklar ditekan minimal 1 kali, dan maksimal 500 kali.

```
4 1                                    h#1: 4
10 3                                   h#2: 4
10 4                                   h#3: 6
```

4.10 Adakah yang Lebih Besar

Ini sebuah permainan angka. Misalnya kita punya angka 324, terdiri dari 3 angka yaitu angka 3, 2, dan 4. Menggunakan angka yang sama, disusun ulang sebuah angka baru. Berapa banyak angka yang bisa disusun yang nilainya lebih besar dari angka semula? Untuk contoh ini, kita lihat kemungkinannya sebagai berikut.

```
324 <-- angka awal
------------------
342     lebih besar
234     lebih kecil
243     lebih kecil
423     lebih besar
432     lebih besar
------------------
```

Jawabnya adalah 3, yaitu 342, 423, dan 432.

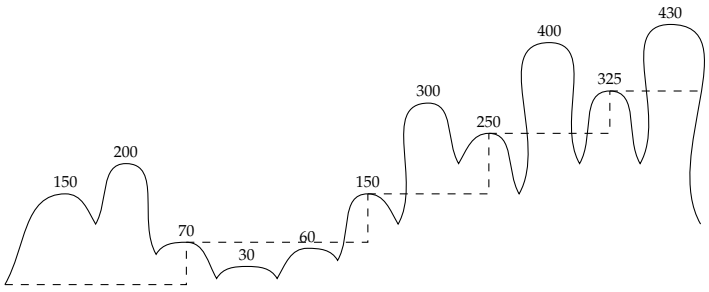
Data soal. Setiap baris terdiri dari satu buah angka. Minimal bernilai 1, dan maksimal 10^6. Seperti contoh di atas, carilah berapa banyak angka baru yang bisa disusun yang nilainya lebih besar dari angka semula.

```
56                                          h#1: 1
1111                                        h#2: 0
9462                                        h#3: 2
```

4.11 Permadani Terbang

Dongeng. Ya, kali ini kisahnya seperti dongeng. Abu Nawas di sebuah negeri ajaib dengan permadani terbang yang juga ajaib. Konon kabarnya negeri ini mempunyai jalur menuju langit. Bersama gunung-gunung dan permadani terbang itu, mungkin langit bisa dicapai. Begitu impian Abu Nawas.

Permadani ini terbang mengikuti ketinggian gunung yang dilewatinya. Ada banyak puncak gunung berjajar sampai jauh. Ada beberapa keistimewaan sang permadani. Pertama, permadani ini hanya bisa naik, tak bisa turun. Setiap melewati sebuah gunung, jika puncak gunung tersebut tingginya tidak lebih dari kemampuan permadani untuk menanjak, maka permadani akan naik sampai ketinggian puncak gunung. Jika tidak, maka permadani akan terbang terus sesuai ketinggian sekarang. Sampai di ujung jajaran gunung yang dilewati permadani, berapa ketinggian yang berhasil dicapai Abu Nawas bersama permadaninya tersebut? Dengan batas kemampuan untuk menanjak sebesar 100 meter, kita lihat contohnya sebagai berikut. Angka menunjukkan ketinggian gunung dalam meter, berjajar mulai dari kiri ke kanan.



Puncak gunung pertama adalah 150. Permadani tidak bisa menanjak, jadi tetap lanjut ke gunung berikutnya, yaitu dengan ketinggian 200. Ini juga tidak bisa, maka dilewati juga. Pada gunung ke-3 dengan ketinggian 70, permadani bisa menuju puncaknya. Sekarang permadani terbang dengan ketinggian 70, melewati dua gunung yang lebih rendah, yaitu 30 dan 60. Pada gunung selanjutnya, dapat lagi naik sampai ketinggian 150, karena selisih dengan posisi sekarang adalah 80 (yaitu 150 - 70, masih dalam kemampuan permadani). Demikian seterusnya, sampai di ujung perjalanan, Abu Nawas bersama permadaninya bisa mencapai ketinggian 325 meter.

Data soal. Setiap baris berisi data tentang kemampuan menanjak permadani, serta deretan ketinggian gunung yang akan dilewati. Setiap angka dipisahkan oleh satu spasi. Angka pertama adalah batas kemampuan permadani untuk menanjak. Angka-angka selanjutnya adalah tinggi puncak gunung yang akan dilewati. Jumlah puncak gunung yang akan dilewati maksimal sebanyak 1000. Carilah ketinggian akhir Abu Nawas setelah melewati semua gunung tersebut.

```
10 3 7 15 25 33 33 40                h#1: 40
5 10 15 30 31 55 7 10 44             h#2: 0
100 90 30 2 85 80 200 33 54          h#3: 90
```

4.12 Tanggal yang Tak Pasti

Penulisan tanggal dalam sebuah bahasa umumnya sudah ada aturan yang baku. Tapi tak semua orang menulisnya sesuai aturan tersebut. Ada banyak cara untuk menuliskan tanggal, misalnya menggunakan angka dengan susunan tanggal-bulan-tahun seperti 21-2-2012 untuk tanggal 21 Februari 2012. Untuk tanggal yang sama bisa juga ditulis 21-02-2012 atau 21/2/2012 atau 21 FEB 2012, dan sekian kemungkinan lainnya. Kadang-kadang antara bulan tanggal dan tahun tidak selalu dalam urutan yang sama, misalnya tanggal di atas ditulis menjadi 2-21-2012. Dalam kasus ini kita akan berusaha memahami sebuah susunan tanggal apakah dapat ditentukan sebagai sebuah tanggal yang benar.

Pemisah antara tanggal bulan dan tahun dapat berupa satu buah spasi, atau garis miring atau sebuah tanda strip. Tanggal bisa ditulis satu atau dua angka. Bulan juga bisa ditulis satu atau dua angka, atau menggunakan singkatan berikut ini (`jan`, `feb`, `mar`, `apr`, `mei`, `jun`, `jul`, `agu`, `sep`, `okt`, `nov`, `des`). Sedangkan tahun selalu ditulis 4 angka. Posisi tanggal, bulan, tahun bisa di mana saja. Bisa di depan, tengah, atau belakang. Contoh berikut ini untuk melihat apakah sebuah tanggal dapat dipastikan atau tidak.

```
15/2/1998   : Bisa, yaitu 15 Februari 1998
04/05/2010  : Tidak, ada dua kemungkinan, yaitu
              4 Mei 2010 atau 5 April 2010
jan 3 2004  : Bisa, yaitu 3 Januari 2004
2011-08-22  : Bisa, yaitu 22 Agustus 2011
```

Data soal. Setiap baris berisi sebuah tanggal dengan susunan yang tidak seragam, seperti uraian di atas. Tentukanlah apakah tanggal tersebut bisa dipastikan, atau tidak. Hasil dari program adalah kata YA apabila tanggal dapat dipastikan. Kata TIDAK jika tanggal tidak bisa dipastikan.

```
23 mar 2000                    h#1: YA
04/07/2010                     h#2: TIDAK
1965-3-agu                     h#3: YA
```

4.13 Jumlah Jam Kerja

Jam kerja karyawan dalam sebuah tempat kerja bisa dihitung dengan berbagai cara. Salah satu cara sederhana adalah dengan mencatat jam masuk dan jam keluar. Berapa total jam kerja dalam satu hari biasanya melalui perhitungan lanjutan yang disesuaikan dengan aturan tertentu. Untuk soal ini klausul terkait dengan perhitungan jam kerja adalah sebagai berikut.

- Jam kerja normal adalah 8 jam
- Jika lebih dari 8 jam, kelebihan jam dihitung sebagai lembur
- Perhitungan total jam lembur adalah sebagai berikut:
  - Satu jam pertama dikali 1
  - Untuk jam ke 2 dikali 2
  - Setiap jam berikutnya dikali 3

Misalnya seorang karyawan masuk jam 7, keluar jam 19, total jam kerja karyawan pada hari itu adalah :

```
*  19 - 7 = 12 jam
*  jam kerja normal adalah 8 jam.
*  lembur sebanyak 12 - 8 = 4 jam
*  lembur jam pertama dikali satu, jadi 1 x 1 = 1 jam
*  lembur jam ke-2 dikali 2, jadi 1 x 2 = 2 jam
*  lembur berikutnya, sebanyak 2 jam, jadi 2 x 3 = 6 jam
*  total lembur adalah 1 + 2 + 6 = 9 jam
*  total jam kerja hari itu adalah 8 + 9 = 17 jam
```

Data soal. Setiap baris berisi angka jam masuk dan jam keluar karyawan. Pemisah antar angka adalah satu spasi. Jam keluar selalu lebih besar dari jam masuk. Angka jam terkecil adalah 0, dan angka jam terbesar adalah 23. Hitunglah jumlah total jam kerja karyawan sesuai dengan ketentuan yang diuraikan di halaman sebelumnya.

```
7 10                                    h#1: 3
10 20                                   h#2: 11
1 10                                    h#3: 9
```

4.14 Beban untuk Keledai

Kali ini Abu Nawas kembali terlibat. Bersama keledainya, dia pergi belanja ke pasar membeli banyak barang. Barang-barang tersebut dibawa menggunakan dua buah keranjang yang dipanggul oleh keledai. Satu keranjang di kiri, satu lagi di kanan. Agar beban yang dibawa keledai tidak berat sebelah, diusahakan agar beban yang masuk di keranjang sebelah kanan sama dengan beban yang dimasukkan ke keranjang sebelah kiri. Jika tidak bisa sama, maka dicari agar selisih berat kiri dan kanan sesedikit mungkin. Berikut ini contoh daftar berat barang yang akan dimasukkan ke dalam keranjang.

```
-----------------------------------------------------
Daftar berat barang : 10 10 20 15 5 5 10 20 20 20 15 10
-----------------------------------------------------
Keranjang kiri      : 20 20 20 10 5 5          total 80
Keranjang kanan     : 20 15 15 10 10 10        total 80
-----------------------------------------------------
```

Dalam contoh ini kita bisa membuat sama berat antara keranjang kiri dan keranjang kanan, yaitu sama-sama 80. Bagaimana kombinasi pengisian barang tidak menjadi perhatian. Demikian juga halnya dengan jumlah barang yang dimasukan ke kiri atau ke kanan bisa diabaikan. Perhatian kita hanya pada keseimbangan antara berat keranjang kiri dan kanan.

Data soal. Setiap baris berisi daftar berat barang. Setiap angka dipisahkan oleh satu spasi. Jumlah minimal barang adalah 2, dan maksimal 100. Hasil yang akan dicari adalah selisih minimal yang bisa diperoleh antara berat keranjang kiri dan keranjang kanan.

```
40 80 10 10 10                          h#1: 10
5 10                                    h#2: 5
2 2 4 4 6 6 8 8 2 2                     h#3: 0
```

4.15 Dari Kursi Jadi Bui

Ada sebuah kata. Kita sebut saja sebagai kata pertama. Kemudian ada kata lainnya, yang kita sebut sebagai kata kedua. Kata pertama akan ditransformasikan menjadi kata kedua. Proses perubahan dari kata pertama menjadi kata kedua tentu saja melalui sejumlah langkah. Langkah yang dilalui misalnya membuang huruf, menambah huruf, atau menggeser posisi huruf.

Kita akan menghitung berapa langkah yang diperlukan untuk mentransformasikan kata pertama menjadi kata kedua. Langkahnya hanyalah terkait dengan membuang dan menambah huruf. Apabila ada pembuangan huruf sebanyak 3, maka dihitung 3 langkah. Jika ada penambahan 2 huruf, maka dihitung 2 langkah. Pergeseran huruf tidak menjadi hitungan dalam proses ini.

Berikut ini kita lihat contoh perubahan dari kata pertama menjadi kata kedua.

```
-------------------------
Kata pertama       : KURSI
-------------------------
1. Buang huruf K   : URSI
2. Buang huruf R   : USI
3. Buang huruf S   : UI
4. Tambah huruf B  : BUI
-------------------------
Kata kedua         : BUI
-------------------------
```

Dari contoh ini diperlukan 4 langkah untuk mentransformasikan kata KURSI menjadi BUI. Tiga langkah membuang huruf, dan satu langkah menambah huruf.

Data soal. Dalam setiap baris terdiri dari dua kata yang dipisahkan oleh satu spasi. Kata terdiri dari huruf A-Z. Hitunglah jumlah langkah yang diperlukan utuk merubah kata pertama menjadi kata kedua.

```
BAYAM AYAM                                h#1: 1
IKAN SAPI                                 h#2: 4
HITAM PUTIH                               h#3: 4
```

4.16 Pita Angka

Ada sebuah pita angka berisi sejumlah angka acak. Setiap angka adalah unik. Tidak ada angka yang sama dengan angka lainnya. Di atas pita terdapat sebuah pena yang bisa digeser ke kiri dan ke kanan. Awalnya pena berada di atas angka pertama, paling kiri.

Kita akan memberi tanda pada setiap angka menggunakan pena tersebut. Memberi tanda pada angka harus dilakukan dari angka terkecil sampai angka terbesar. Jika awalnya pena tidak berada pada angka paling kecil, maka pena harus digeser terlebih dahulu ke atas angka yang paling kecil. Setelah diberi tanda, pena digeser lagi ke angka berikutnya, sesuai urutan dari kecil ke besar. Demikian dilakukan sampai semua angka diberi tanda. Berapa jauh pena digeser jika jarak antar angka adalah 10 cm? Mari kita lihat contoh berikut ini.

```
4   8   2   1   5   7   9
^--- --- --- --- --- ---   Posisi awal pena
 --- --- ---^--- --- ---   Ke angka terkecil  1   30cm
 --- ---^--- --- --- ---   Angka selanjutnya  2   10cm
^--- --- --- --- --- ---   Angka selanjutnya  4   20cm
 --- --- --- ---^--- ---   Angka selanjutnya  5   40cm
 --- --- --- --- ---^---   Angka selanjutnya  7   10cm
 ---^--- --- --- --- ---   Angka selanjutnya  8   40cm
 --- --- --- --- --- ---^  Angka selanjutnya  9   50cm
==================================== Total === 200cm
```

Dari ilustrasi di atas didapat total jarak pergeseran pena adalah 200cm.

Data soal. Setiap baris berisi angka acak. Pemisah setiap angka adalah satu spasi. Jumlah angka minimal adalah 10, dan maksimal 1000. Jarak setiap angka adalah 10cm. Hitung jarak pergeseran pena seperti pada penjelasan di atas.

```
1 2 3 4 5 6 7 8 9                    h#1: 80
40 50 20 21 22 23 24 25              h#2: 150
100 95 90 80 40 10 8                 h#3: 120
```

4.17 Memburu Harta Karun

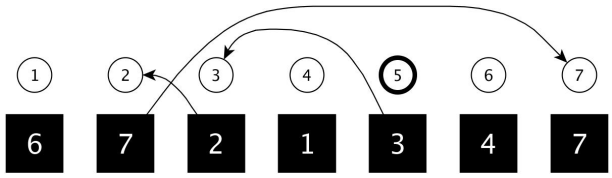
Novel Da Vinci Code menggunakan alur seperti soal ini. Eh tunggu. Mungkin bisa juga sebaliknya, soal ini seperti alur dalam kisah novel tersebut. Perburuan harta karun adalah cerita dengan lompatan dari satu petunjuk ke petunjuk lainnya. Tentu saja akhirnya adalah penemuan harta yang diburu. Namun tak jarang hasilnya adalah harapan kosong.

Kasus ini adalah sebuah model yang sama dengan cerita di atas. Sebuah kotak jika dibuka, berisi petunjuk yang akan membawa kita ke kotak berikutnya. Kotak berikutnya ini juga berisi informasi untuk menuju ke kotak lain. Demikian seterusnya sampai di kotak terakhir yang berisi harta karun.

Isi kotak adalah sebuah angka. Misalnya angka 5. Angka ini menunjukkan bahwa kita harus menuju ke kotak dengan nomor 5. Kotak nomor 5 kita buka, isinya juga sebuah angka lagi, misalnya 9. Kita menuju ke kotak 9, untuk melihat lagi angka yang ada di dalamnya. Demikian seterusnya menelusuri kotak demi kotak. Apabila kita menemukan kotak yang berisi angka sama dengan nomor kotak tersebut, maka di sanalah pencarian berakhir. Kotak itu berisi harta karun. Demikianlah aturannya. Berikut ini sebagai ilustrasinya.

```
Langkah 1 : Buka kotak nomor 5 : Berisi angka 3
Langkah 2 : Buka kotak nomor 3 : Berisi angka 2
Langkah 3 : Buka kotak nomor 2 : Berisi angka 7
Langkah 4 : Buka kotak nomor 7 : Berisi angka 7
```

Mulainya dari kotak yang mana? Diberikan nomor kotak awal untuk pencarian. Untuk contoh ini kita mulai dari kotak nomor 5. Seperti diagram di samping, kita membuka 4 kotak untuk sampai pada kotak berisi harta karun, yaitu kotak nomor 7.



Apakah pencarian selalu menemukan sebuah kotak dengan angka yang sama dengan nomor kotaknya? Tidak. Dalam hal ini, artinya kita tidak berhasil menemukan kotak yang berisi harta karun, meskipun kotak itu sesungguhnya ada. Atau memang tidak ada kotak yang berisi harta karun.

Dari diagram yang sama, jika kita mulai dari kotak nomor 6, maka tidak akan pernah menemukan harta karun, karena akan berputar-putar pada tiga kotak saja. Dari kotak nomor 6 ke kotak nomor 4. Kemudian ke ke kotak nomor 1. Dari kotak nomor 1 kembali lagi ke kotak nomor 6. Kotak harta karun ada dalam daftar di atas, sayangnya langkah kita tidak mengantarnya ke sana.

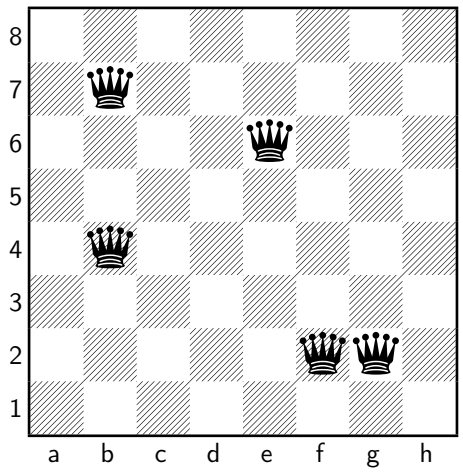
Data soal. Setiap baris berisi sejumlah angka. Setiap angka dipisahkan oleh satu spasi. Angka pertama adalah nomor kotak yang akan dibuka pertama kali. Angka ke-2 adalah isi dari kotak nomor 1, angka ke-3, adalah isi dari kotak nomor 2, demikian selanjutnya sampai angka terakhir pada baris tersebut. Adakalanya sebuah kotak berisi angka 0, atau lebih dari jumlah kotak yang ada. Jika ditemukan seperti ini, artinya harta karun tidak akan bisa ditemukan. Hasil dari program adalah jumlah kotak yang dibuka untuk sampai pada kotak yang berisi harta karun. Jika tidak ditemukan harta karun, maka tampilkan huruf X.

```
1 3 1 2 4                       h#1: X
5 9 11 4 7 5 2 2 0              h#2: 1
2 1 3 1 5 9 12 77 123           h#3: 3
```

4.18 Menteri yang Saling Ancam

Papan catur terdiri dari 8 kali 8 kotak, yaitu 8 kolom dan 8 baris. Kotak berwarna hitam putih berselang-seling. Dalam permainan catur ada berbagai buah catur dengan gerakan tertentu. Kita akan memilih satu buah catur saja, yaitu menteri. Menteri bisa bergerak lurus. Bisa ke depan, ke belakang, ke kiri, ataupun ke kanan. Jauh gerakan tidak terbatas, tentunya tetap di dalam papan catur. Selain itu, menteri juga bisa bergerak secara diagonal. Sama seperti gerakan lurus, gerakan diagonal juga tak terbatas, bisa ke arah diagonal yang mana saja.

Setiap baris dan kolom pada papan catur diberi label. Untuk kolom diberi label huruf `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h`. Sedangkan baris diberi label dengan angka, yaitu `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`. Setiap kotak pada papan catur dapat dirujuk atau diberi label dengan kombinasi kolom dan baris. Misalnya kotak yang berada pada kolom ke-3 baris ke-5 diberi label **c5**.



Pada papan catur ini akan ditempatkan menteri sebanyak lebih dari 1. Misalnya kita akan letakan 5 buah menteri seperti pada diagram di samping. Kita akan mencari apakah ada menteri yang posisinya dapat mengancam menteri lainnya.

Pertama kita lihat menteri yang ada di kotak b4 dan b7. Kedua menteri ini saling ancam karena berada dalam kolom yang sama. Kemudian menteri yang ada di kotak f2 dan g2. Kedua menteri ini juga saling ancam, karena berada pada baris yang sama. Menteri yang ada di e6 tidak mengancam menteri yang lain, baik melalui baris, kolom, atau diagonal, tidak ada yang melewati jalur yang sama. Untuk menteri yang ada di kotak g2 dan b7, juga saling ancam melalui jalur diagonal. Dengan melihat susunan dalam diagram ini, dapat diketahui, dari 5 buah menteri yang ditempatkan di papan catur, ada 4 menteri yang posisinya bisa saling ancam.

Data soal. Setiap baris berisi data kotak mana saja yang berisi menteri. Data berupa label dari kotak yang berisi menteri. Setiap label dipisahkan oleh satu buah spasi. Setiap baris berisi minimal 2 label, maksimal 64 label. Hasil dari program adalah jumlah menteri yang posisinya saling ancam.

```
a1 b7                                   h#1: 0
c1 d2 e3 f4                             h#2: 4
b2 b3 g1                                h#3: 2
```

4.19 Saklar Seven Segment

Pernah melihat tampilan angka yang menggunakan *seven segment*? Biasanya digunakan pada alat elektronik sekitar tahun 80-an. Sekarang di era *cyberspace* tentu masih ada yang memanfaatkannya. Misalnya jam digital, tampilan waktu di lampu pengatur lalu lintas, *speedometer* kendaraan bermotor, dan peralatan elektronik lainnya.

Seven segment terdiri dari garis vertikal dan horizontal yang disusun membentuk angka 8. Setiap garis diberi lampu yang bisa dinyalakan atau dimatikan. Bila lampu menyala semua, maka akan menghasilkan tampilan angka 8. Berikut ini disajikan dalam sebuah diagram. Disertai tabel angka 0 sampai 9 beserta data lampu nomor berapa saja yang menyala untuk menampilkan angka tersebut.

```
-------------------------
Angka  1  2  3  4  5  6  7 <- Nomor lampu
-------------------------
  0    1  2  3  4  5  -  7    Tabel lampu yang menyala
  1    -  2  -  4  -  -  -    untuk sebuah angka.
  2    -  2  3  -  5  6  7    Tanda strip menunjukkan
  3    -  2  -  4  5  6  7    lampu dalam kondisi mati
  4    1  2  -  4  -  6  -
  5    1  -  -  4  5  6  7
  6    1  -  3  4  5  6  7
  7    1  2  -  4  5  -  -
  8    1  2  3  4  5  6  7
  9    1  2  -  4  5  6  7
-------------------------
```

Dengan kondisi awal semua lampu dalam keadaan mati, kita akan menampilkan sederetan angka. Kita hanya memiliki satu buah penampil *seven segment*. Artinya, angka-angka ini akan ditampilkan bergantian. Misalnya akan menampilkan 3 deretan angka, yaitu 5, 7, dan 1. Kondisi akhirnya, semua lampu kembali mati. Berapa kali kita harus menyalakan/mematikan saklar untuk menampilkan deretan angka ini? Setiap lampu mempunyai satu saklar.

```
--------------------
       1 2 3 4 5 6 7   <- Nomor lampu
--------------------
Angka  - - - - - - -
  5    1 - - 4 5 6 7                    5 kali menyalakan
  7    1 2 - 4 5 - -   2 kali mematikan, 1 kali menyalakan
  1    - 2 - 4 - - -   2 kali mematikan
       - - - - - - -   2 kali mematikan
--------------------
```

Total menyalakan saklar sebanyak 6 kali, dan 6 kali mematikan. Jadi perlu 12 kali menyalakan/mematikan saklar untuk menampilkan deretan angka 5, 7, dan 1. Jika ada angka sama berurutan, maka sebelum menampilkan angka berikutnya, semua lampu harus dimatikan terlebih dahulu, baru angka berikutnya ditampilkan. Misalnya angka 1 yang diikuti angka 1 lagi. Pertama dinyalakan dua lampu. Karena angka berikutnya sama, maka dimatikan semua lampu dulu, yaitu ada dua. Kemudian baru menampilkan angka berikutnya yaitu angka sama (angka 1), dengan menyalakan dua buah lampu. Untuk contoh terakhir ini, diperlukan 4 kali menyalakan dan 4 kali mematikan lampu.

Data soal. Setiap baris terdiri dari beberapa angka yang dipisahkan oleh satu spasi. Angka dalam rentang 0 sampai 9. Jumlah angka dalam satu baris minimal 1, maksimal 1000. Hitunglah berapa kali menyalakan dan mematikan saklar untuk menampilkan deretan angka tersebut pada sebuah display *seven segment*. Kondisi awal adalah semua lampu mati, dan kondisi akhir, semua lampu kembali mati.

```
5 6                                     h#1: 12
1 1                                     h#2: 8
0 8                                     h#3: 14
```

4.20 Minesweeper Satu Dimensi

Minesweeper adalah sebuah permainan. Saya pertama kali mengenalnya dalam Windows 95. Permainan ini berupa kotak-kotak yang disusun dalam sejumlah baris dan kolom. Kotak dalam keadaan tertutup. Di dalamnya ada petunjuk berupa angka, ada juga yang kosong, ada kotak yang berisi ranjau. Kalau kotak yang dibuka berisi ranjau, maka permainan berakhir, dan pemain kalah. Jika berhasil menentukan semua kotak berisi ranjau tanpa membukanya, maka pemain berhasil memenangkannya.

Kita akan menggunakan ide permainan ini, tetapi disederhanakan. Jika minesweeper terdiri dari baris dan kolom, maka dalam soal ini hanya satu dimensi saja. Sejumlah kotak disusun dalam satu baris. Berikut ini diagram untuk minesweeper satu dimensi dalam keadaan semua kotak terlihat isinya.

1	R	1	0	0	1	R	2	R	1	0	0

Kotak yang berisi huruf R adalah kotak yang berisi ranjau. Kotak yang berisi angka 0 menunjukkan tidak ada kotak yang berisi ranjau di sebelahnya. Kotak yang berisi angka 1 menunjukkan bahwa ada satu buah kotak yang berisi ranjau di sebelahnya. Bisa di sebelah kiri atau di sebelah kanan. Sedangkan kotak yang berisi angka 2 menandakan kotak yang ada di sebelah kiri dan sebelah kanannya berisi ranjau.

Sekarang persoalan kita adalah minesweeper satu dimensi ini kehilangan semua kotak yang berisi ranjau. Yang tersisa hanya kotak yang berisi angka. Agar kembali menjadi lengkap, perlu disisipkan sejumlah kotak yang berisi ranjau. Berapa kotak ranjau diperlukan? Kita akan lihat dalam contoh berikut ini.

0	1	1	0	1	2	1	0	1

Minesweeper yang kehilangan kotak ranjau

0	1	R	1	0	1	R	2	R	1	0	1	R

Setelah disisipkan kotak ranjau di posisi yang benar

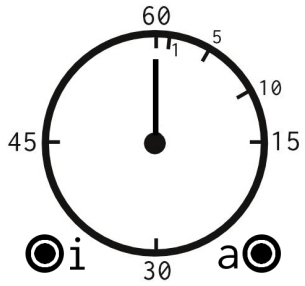
Dari contoh ini, diperlukan 4 kotak ranjau untuk menjadikan minesweeper kembali ke komposisi yang benar.

Data soal. Setiap baris berisi sejumlah angka. Setiap angka dipisahkan oleh satu spasi. Angkanya adalah 0, 1, dan 2. Satu baris angka ini menunjukkan minesweeper yang kehilangan kotak ranjau. Program dibuat untuk mecari berapa kotak ranjau yang diperlukan agar minesweeper menjadi lengkap.

```
0 1 1 0 0 0 0                           h#1: 1
1 0 0 0 0 0 1 2 1 0                     h#2: 3
0 0 0 0 0 0 0 0 0 0                     h#3: 0
```

4.21 Detik Tanpa Menit

Sebuah jam dinding. Hanya memiliki jarum panjang yang biasa dipakai untuk menunjukkan detik. Tidak ada jarum menit. Tidak ada jarum pendek yang menunjukkan jam. Angka yang tertera di sekeliling jam ini adalah dari 1 sampai 60, sesuai jumlah detik dalam satu menit. Di sebelah kirinya ada sebuah tombol. Demikian juga di sebelah kanan ada sebuah tombol. Jika tombol kiri ditekan, maka jarum bergerak satu detik berlawanan dengan arah jarum jam. Begitu juga dengan tombol kanan, kalau ditekan, jarum bergerak satu detik searah jarum jam.



Ada perilaku jarum ini yang perlu diketahui lagi. Jarum baru mulai bergerak jika tombol yang sama sudah ditekan lebih dari satu kali berturutan. Kalau hanya ditekan sekali, jarum tidak bergerak. Ditekan lagi, baru jarum bergerak satu detik sesuai dengan arah tombol mana yang ditekan. Ditekan lagi tombol yang sama, jarum bergerak lagi satu detik. Demikian seterusnya.

Untuk tombol kiri diberi kode huruf **i**, sedangkan tombol kanan diberi kode **a**. Pada awalnya jarum menunjuk ke atas berada di angka 60. Setelah tombol ditekan seperti pola berikut, di angka berapakah jarum berhenti?

```
      a  a  a  i  a  i  a  i  i  i  i  a  a  a  i
-------------------------------------------------
60    60 1  2  2  2  2  2  2  1  60 59 59 60 1  1
```

Posisi awal jarum menunjuk angka 60. Ditekan a sekali, jarum belum bergerak, masih di angka 60. Setelah ditekan lagi, baru jarum bergerak ke angka 1. Demikian seterusnya sesuai deskripsi di atas tentang jam ajaib ini. Dari diagram, posisi akhir jarum menunjuk ke angka 1 setelah tombol ditekan seperti pola yang dicontohkan di atas.

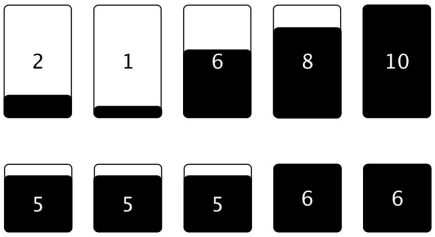
Data soal. Setiap baris berisi susunan huruf a dan i. Tidak ada pemisah antara huruf. Minimal terdiri dari 1 huruf, dan maksimal 1000 huruf. Tentukanlah posisi terakhir jarum setelah tombol ditekan seperti pola yang ada di setiap baris soal.

```
aiaiaiaiaiaiaiai                    h#1: 60
aaaaaaaaaaaaaaaaaaaaa               h#2: 21
aaiiaa                              h#3: 1
```

4.22 Ember yang Seragam

Di sebuah negeri yang tak bernama, ada aturan tentang ember. Jika membeli ember lebih sari satu, harus membeli satu jenis saja. Ukurannya sama, warnanya sama. Tak boleh beda-beda. Kali ini kita melibatkan Nasrudin Hoja. Dia punya banyak ember. Tentu saja semuanya sama. Semuanya berisi air dengan volume yang berlainan. Nasrudin berfikir, alangkah baiknya jika semua ember ini bisa disamakan isinya, sehingga dia bisa mengganti embernya dengan yang lebih kecil. Lagi-lagi semua ember penggantinya tentu saja harus sama.

Bagaimana cara menyeimbangkan isi ember-ember tersebut? Ada lagi satu syarat. Isinya hanya bisa dipindah per satu liter. Nasrudin punya dua ember dengan kapasitas 10 liter. Ember A berisi 10 liter, ember B berisi 6 liter. Maka isi ember A bisa diambil 2 liter untuk dimasukkan ke ember B. Hasilnya adalah ember A jadi 8 liter, ember B juga 8 liter. Dengan demikian, Nasrudin bisa mengganti embernya menjadi yang 8 liter saja, dibanding yang sebelumnya yang lebih besar. Itu untuk kasus sederhana 2 buah ember. Kita akan lihat contoh berikut ini, untuk 5 buah ember. Jumlah ember penggantinya harus sama dengan jumlah ember semula.



Dari contoh di samping, awalnya perlu ember minimal dengan kapasitas 10 liter. Setelah isinya dipindah, cukup menggunakan ember yang 6 liter.

Data soal. Setiap baris berisi sejumlah angka. Angka ini menunjukkan isi sebuah ember. Setiap angka dipisahkan oleh satu spasi. Jumlah angka, menunjukkan jumlah ember. Maksimal jumlah ember adalah 1000. Isi ember minimal 1 liter, dan maksimal 50 liter. Carilah ukuran ember yang paling kecil yang mungkin untuk menampung isi air.

```
2 2 2 2                                  h#1: 2
10 12 8 2 2 2 2 2                        h#2: 5
20 10                                    h#3: 15
```

4.23 Kata Dalam Mantra

Ada 10 ribu huruf. Terletak dalam 100 kolom sebanyak 100 baris. Hurufnya tak beraturan. Kalaupun bisa dibaca, mungkin berbunyi seperti mantra. Di dalam ramainya huruf ini kita akan cari sebuah kata, mungkin saja terselip di dalamnya. Mungkin kata itu tersusun mendatar, mungkin juga vertikal. Arahnya bisa dari kiri ke kanan, bisa dari kanan ke kiri, bisa dari atas ke bawah, bisa juga dari bawah ke atas. Tak usah repot-repot mencari secara diagonal, karena itu menambah kesulitan. Kita buat saja contohnya agar lebih jelas. Cukup 10 kolom kali 10 baris. Lalu cari kata **MANUSIA** di dalamnya.

```
D  W  J  B  Y  O  P  E  Q  S
V  V  H  I  U  E  S  Z  L  K
E  W  S :M  V  N  K  Y  G  D
S  Z  C :A  F  B  N  F  J  K
E  E  S :N  F  X  A  W  T  E
G  B  N :U  O  P  L  A  R  V
F  C  B :S  W  E  Y  U  J  M
A  Z  G :I  F  D  D  V  B  C
N  H  J :A  V  D  F  G  W  E
D  S  A  T  G  B  H  J  K  L
```

Data soal. Setiap baris terdiri dari dua data yang dipisahkan oleh satu spasi. Data pertama berupa kata yang akan dicari dalam kumpulan huruf. Data kedua, berupa 10 ribu huruf. Huruf ini harus disusun agar menjadi 100 kolom kali 100 baris. Dari huruf pertama sampai huruf ke 100 berada di baris pertama. Huruf ke-101 sampai huruf 200 berada di baris ke-2. Demikian seterusnya sampai baris ke 100 yang terdiri dari huruf ke-9901 sampai 10000. Hasil dari program adalah kata ADA atau TIDAK. Kata ADA ditampilkan jika ditemukan kata dalam kumpulan huruf tersebut. Kata TIDAK apabila tidak ditemukan. Contoh di bawah ini hanyalah untuk ilustrasi, karena keterbatasan ruang untuk menampilkan 10 ribu huruf.

```
MELATI DJNKMELATIJFN...              h#1: ADA
MAWAR KLASDMJNFJOUIEFDSWEG...        h#2: TIDAK
MATAHARI GFHBSNWTUPOQPKASBXAMS...    h#3: TIDAK
```

4.24 Kunci di Ujung Jari

Dunia makin maju. Membuka kunci tak melulu urusan anak kunci. Ada *password*. Ada juga melalui sidik jari. Yang lebih tinggi lagi teknologinya pakai pola iris mata. Mari lihat layar sentuh. Banyak dipakai di telepon seluler. Untuk membuka kunci aktivasi, ada yang disebut dengan *pattern lock*. Terdiri dari 9 titik. Disusun dalam tiga baris dan tiga kolom. Titik ini harus dihubungkan dengan sebuah goresan, bisa pakai ujung jari. Jalur goresan yang benar akan membuka kunci.

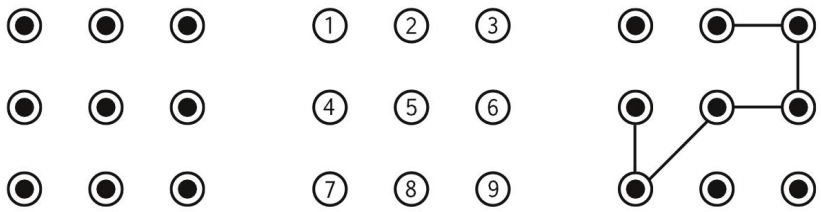


Diagram paling kiri menunjukkan susunan sembilan buah titik. Sembilan angka yang di tengah adalah representasi titik dalam angka. Sedangkan yang paling kanan adalah sebuah bentuk pola jalur ketika membuka kunci. Pola ini dimulai dari tengah atas, dan berakhir di titik tengah sebelah kiri. Jika pola ini ditulis dalam bentuk angka, maka urutannya adalah 2, 3, 6, 5, 7, 4.

Ada beberapa aturan sebuah kunci yang dibolehkan.

- Minimal melalui 5 titik
- Maksimal melalui 7 titik
- Dari satu titik harus ke titik terdekat. Titik terdekat adalah titik yang ada di sampingnya, baik kiri, kanan, atas, bawah, atau miring. Misalnya dari titik 1 bisa ke titik 2, 4, dan 5. Dari titik 1 tidak boleh ke titik 3, 6, 7, 8, dan 9.
- Boleh melewati titik yang sama lebih dari sekali.

Data soal. Setiap baris berisi angka yang merupakan urutan pola kunci. Periksalah pola ini apakah valid sesuai dengan persyaratan di atas. Hasilnya adalah kata YA untuk pola yang valid, kata TIDAk untuk pola yang tidak valid.

```
12321                                   h#1: YA
23654789                                h#2: TIDAK
512369                                  h#3: YA
```

4.25 Dua Tanggal

Deskripsi soal ini ringkas. Hitunglah jumlah hari antara dua buah tanggal. Hari pada kedua tanggal yang diberikan masuk dalam hitungan. Misalnya tanggal pertama adalah 2 Mei 2010, tanggal kedua adalah 4 Mei 2010. Jumlah hari antara kedua tanggal ini adalah 3 hari, yaitu 2 Mei 2010, 3 Mei 2010 dan 4 Mei 2010. Tanggal pertama tidak harus lebih kecil dari tanggal kedua. Tanggal pertama bisa saja lebih besar dari tanggal pertama, atau kedua tanggal tersebut sama. Berikut beberapa contoh.

```
-----------------------------------------------------------
14-02-1990 20-02-1998  7 hari  Tanggal pertama lebih kecil
30-01-2000 10-01-2000 21 hari  Tanggal pertama lebih besar
30-01-1980 02-02-1980  4 hari  Tanggal pertama lebih kecil
11-05-2010 11-05-2010  1 hari  Tanggal sama
-----------------------------------------------------------
```

Data soal. Setiap baris berisi dua buah tanggal yang dipisahkan oleh satu spasi. Format tanggal adalah tanggal-bulan-tahun. Pemisah antara tanggal, bulan, dan tahun adalah satu buah karakter strip (-). Tanggal dan bulan terdiri dari dua angka, sedangkan tahun terdiri dari 4 angka. Hasil dari program adalah jumlah hari antara dua tanggal.

```
01-01-2000 01-02-2000                h#1: 32
02-02-1999 02-02-1999                h#2: 1
10-10-2010 01-10-2010                h#3: 10
```

4.26 Menuju Nol

Saya mencari hubungan antara *Multi Level Marketing (MLM)* dengan *Zero Sum Game* melalui Google. Sayang hasilnya tak memenuhi harapan. Mungkin gagal memberikan kata kunci yang tepat untuk jadi bahan pencarian. Tapi tunggu dulu. Mengapa saya menulis tentang MLM? Mengapa harus ada hubungannya dengan *Zero Sum Game*? Ah, anggap saja masih punya relasi dengan soal ini. Meskipun hanya sebiji sawi.

Ini tentang sebuah kemungkinan dan strategi menyusun angka. Ada sejumlah angka, misalnya 10, 20, 15, 10, dan 5. Dengan berbekal tanda plus dan minus yang disisipkan diantara angka-angka ini, apakah totalnya bisa menjadi nol? Posisi tanda minus dan tanda plus bisa dimana saja diantara angka tersebut. Selain itu, susunan angkanya boleh saja diatur lagi dengan urutan yang berbeda. Untuk lima angka ini bisa menjadi nol, yaitu disusun sebagai berikut.

```
10  +  20  -  15  -  10  -  5   <-- seperti urutan asli
20  -  15  -   5  +  10  - 10   <-- urutannya diganti
```

Data soal. Setiap baris berisi sejumlah angka yang dipisahkan oleh satu spasi. Banyaknya angka dalam satu baris minimal 2, dan maksimal 100 angka. Nilai angka minimal 0, dan maksimal 1000. Dengan menyisipkan tanda plus dan minus diantara angka-angka itu, tentukanlah apakah totalnya bisa menjadi nol atau tidak. Hasil dari program adalah kata YA apabila total angka bisa menjadi nol, dan TIDAK jika tidak bisa menjadi nol.

```
5 5                                    h#1: YA
6 3 100                                h#2: TIDAK
0 0 0 20 5 15                          h#3: YA
```

4.27 Nilai Pilihan Ganda

Salah satu bentuk soal dalam ujian adalah soal pilihan ganda. Ada kalimat berupa pertanyaan (mungkin juga pernyataan). Di bawahnya disediakan beberapa pilihan jawaban. Ada 3 pilihan, 4 pilihan, atau 5 pilihan. Diluar itu mungkin jarang orang membuatnya. Cara memberi nilai untuk soal seperti ini mempunyai banyak alternatif. Ada yang memberi nilai jika jawabannya benar, dan tidak ada resiko jika jawabannya salah. Ada juga yang menyebabkan pengurangan nilai jika jawabannya salah.

Untuk soal ini kita akan membuat aturan, setiap jawaban yang benar mendapatkan nilai tertentu, misalnya 5. Apabila menjawab, tapi salah, maka dikurangi dengan nilai tertentu, misalnya 2. Jika tidak menjawab, tidak ada penambahan atau pengurangan nilai. Dengan aturan seperti ini, kita akan menentukan, apakah nilai seseorang benar atau salah. Maksudnya begini. Si Fulan ikut ujian. Jumlah soalnya 20. Kriteria penilaian, seperti uraian di atas. Nilai yang diperoleh si Fulan adalah 68. Apakah nilai 68 mungkin terjadi? Mari kita lihat perhitungan berikut.

```
14 Benar, 1 Salah, 6 Tidak dijawab
(14 x 5) + (1 x -2) + (6 x 0) = 68
```

Nilai 68 adalah mungkin terjadi dari hasil ujian tersebut. Tetapi Fulan tidak mungkin mendapatkan nilai 93, karena tidak ada kombinasi yang memungkinkan adanya nilai 93 jika aturannya seperti di atas.

Data soal. Setiap baris terdiri dari 4 angka. Setiap angka dipisahkan oleh satu spasi. Angka pertama adalah nilai yang diperoleh dari hasil ujian. Angka kedua adalah jumlah soal. Angka ketiga adalah nilai untuk satu jawaban apabila benar. Angka keempat adalah besarnya pengurangan untuk satu jawaban yang salah. Tidak ada penambahan atau pengurangan nilai jika tidak menjawab. Periksalah, apakah nilai hasil ujian mungkin terjadi sesuai dengan angka yang diberikan. Hasil dari program adalah kata YA apabila memang mungkin nilai tersebut didapatkan, dan kata TIDAK jika nilai tersebut tidak mungkin didapatkan.

```
80 10 5 1                                   h#1: TIDAK
90 100 1 1                                  h#2: YA
99 10 10 5                                  h#3: TIDAK
```

4.28 Angka yang Sama

Ada dua himpunan yang berisi sejumlah angka. Setiap himpunan berisi angka yang unik. Tidak ada angka yang sama dalam satu himpunan. Kita akan mencari berapa banyak angka yang sama antara himpunan pertama dengan himpunan kedua. Dalam bahasa matematika, kita akan mencari irisan antara himpunan A dan himpunan B, atau ditulis dalam notasi $A \cap B$.

```
A = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
B = {1, 2, 5, 11, 12, 13, 14, 15, 16, 17}
A irisan B adalah {1, 2, 5}
```

Dari contoh di atas terdapat 3 angka yang sama antara himpunan A dengan himpunan B.

Data soal. Setiap baris berisi sejumlah angka. Angka tersebut merupakan elemen dari himpunan A dan himpunan B. Pemisah antara himpunan A dengan himpunan B adalah sebuah tanda strip (-). Sedangkan setiap angka dipisahkan oleh satu spasi. Hasil dari program adalah jumlah angka yang sama antara himpunan A dan himpunan B.

```
10 11 12 13-14 15 16 17              h#1: 0
1 2 3 4 5 6 7-1 2 3 4 5 6 7          h#2: 7
22 23 45 46-22 23 89 905             h#3: 2
```

4.29 Selisih Terbesar

Kembali menggunakan dua himpunan, kali ini kita akan mencari total selisih terbesar. Himpunan A dan himpunan B mempunyai jumlah elemen yang sama. Kita ambil masing-masing satu elemen dari himpunan A dan himpunan B. Hitung selisihnya. Lakukan lagi untuk masing-masing satu elemen di kedua himpunan. Selisih dari setiap elemen ini kita jumlahkan, sehingga didapat total selisih. Setiap elemen yang ada di himpunan A dan himpunan B hanya boleh digunakan sekali. Kita akan mencari total selisih terbesar yang bisa didapatkan. Contohnya sebagai berikut.

```
A = { 1, 2, 3, 4}
B = {10, 2, 8, 1}
```

```
---------------------------
Elemen    Elemen    Selisih
A         B
---------------------------
1         10              9
2         8               6
3         2               1
4         1               3
---------------------------
Total selisih            19
---------------------------
```

Dengan menggunakan kombinasi pasangan elemen A dengan elemen B seperti diagram di atas, kita dapatkan total selisih terbesar, yaitu 19. Jika menggunakan kombinasi lain akan menghasilkan total selisih yang lebih kecil.

Data soal. Setiap baris berisi data himpunan A dan himpunan B. Data berupa angka yang merupakan elemen dari himpunan. Pemisah antara himpunan A dengan himpunan B adalah sebuah tanda strip (-). Sedangkan setiap angka dipisahkan oleh satu spasi. Hasil dari program adalah total selisih terbesar yang bisa didapatkan dari kedua himpunan tersebut.

```
1 2 3 4-1 2 3 4                          h#1: 8
10 20 30-0 0 0                           h#2: 60
5 5 1-0 1 2                              h#3: 10
```

4.30 Angka yang di Tengah

Mari bicara angka lagi. Ah, kalimat yang salah. Dalam buku ini, tak ada soal yang beranjak dari angka. Semuanya tentang angka. Jadi, mari susun sejumlah angka. Dari angka awal sampai angka akhir. Berurutan. Ketika menulis urutan angka itu jangan beri pemisah. Sambungkan saja semuanya. Lihat contoh berikut dengan angka awal 10 dan angka akhir 30. Contoh kedua dengan angka awal 1 dan angka akhir 21.

```
10111213141516171819202122324252627282930
--------------------==--------------------
123456789101112131415161718192021
----------------=----------------
```

Apakah yang hendak dicari? Kita akan mencari angka yang berada di tengah. Kalau panjang angkanya ganjil, ambil satu angka yang tepat di tengah. Kalau panjangnya genap, ambil dua angka yang berada di tengah. Untuk contoh yang pertama, panjang angkanya adalah 42. Artinya kita akan mengambil 2 angka yang berada di tengah, yaitu angka ke-21 dan angka ke-22. Hasilnya adalah angka 20. Sedangkan contoh kedua, panjangnya adalah 33, angka yang di tengah adalah satu angka di posisi ke-17 yaitu angka 3.

Data soal. Setiap baris berisi 2 angka yang dipisahkan oleh satu spasi. Angka pertama adalah angka awal, dan angka kedua adalah angka akhir. Angka minimal adalah 1, dan maksimal adalah 10^6. Dengan menyusun angka awal sampai angka akhir berurutan, tentukanlah angka yang berada di posisi tengah.

```
1 3                                     h#1: 2
100 104                                 h#2: 0
40 44                                   h#3: 42
```

4.31 Tempat Terbatas

Ada sebuah alat. Fungsinya untuk melakukan operasi matematika. Hanya ada 3 operasi yang bisa dilakukan alat ini, yaitu penjumlahan menggunakan tanda +, pengurangan menggunakan tanda -, dan perkalian menggunakan tanda *. Keterbatasan berikutnya, pada satu waktu, alat ini hanya bisa melakukan operasi untuk dua angka dengan satu operator.

Hasil operasi disimpan kembali ke tempat yang sama, dan menjadi angka pertama untuk operasi selanjutnya jika ada. Angka yang akan diproses oleh alat ini disusun dengan cara tertentu. Angka dan operator disusun berselang-seling. Data pertama adalah angka, kemudian operator, lalu angka kedua, selanjutnya operator kedua, angka lagi, dan seterusnya. Data terakhir adalah sebuah angka. Dari diagram di samping, hasil dari proses yang dilakukan alat terhadap *input* adalah 200.

```
Data yang akan diproses
3 + 2 - 6 + 100 * 2 + 2
-----------------------
Langkah  Operasi  Hasil
-----------------------
1         3 + 2        5
2         5 - 6       -1
3        -1 + 100     99
4        99 * 2      198
5       198 + 2      200
-----------------------
```

Data soal. Setiap baris berisi angka dan operator. Setiap angka dan operator dipisahkan oleh satu spasi. Angka minimal adalah 0, dan angka maksimal adalah 100. Jumlah operator minimal adalah 1 operator, dan maksimal adalah 99 operator. Hasil dari program adalah angka akhir yang dihasilkan mesin ini setelah memproses data yang diberikan. Apabila hasil akhir kecil dari nol, tampilkan nilai absolutnya.

```
4 + 2 - 0 * 0                    h#1: 0
1 + 1 + 1 + 1                    h#2: 4
10 - 10 * 100 - 5                h#3: 5
```

4.32 Menghitung Palindrom

Bilangan palindrom[3] punya sifat yang menarik. Bilangan ini jika dibaca dari kiri ke kanan, atau kanan ke kiri, mempunyai nilai yang sama. Misalnya bilangan 212. Bilangan ini dibaca dari kedua arah menghasilkan nilai yang sama, yaitu 212. Bandingkan misalnya dengan bilangan 365, jika dibaca dari kanan ke kiri menjadi 563. Tidak sama. Ada banyak bilangan palindrom. Kita akan mencari ada berapa bilangan palindrom dalam rentang tertentu. Ambil contoh dari bilangan 100 sampai dengan 150. Ada 5 bilangan palindrom, yaitu :

```
101   111   121   131   141
```

Data soal. Setiap baris berisi dua angka yang dipisahkan oleh satu spasi. Kita akan mencari ada berapa banyak bilangan palindrom yang ada diantara kedua angka tersebut. Kedua angka disertakan dalam rentang yang dimaksud. Angka minimal 1 dan maksimal 10^9. Angka kedua selalu lebih besar dari angka pertama. Hasil dari program adalah jumlah bilangan palindrom yang ada dalam rentang tersebut.

```
1 10                                    h#1: 9
99 100                                  h#2: 1
21 31                                   h#3: 1
```

[3] **palindrom** *n Ling* kata, rangkaian kata, atau bilangan yg terbaca sama, baik dr depan maupun dr belakang, spt kodok, radar, taat — Kamus Besar Bahasa Indonesia, edisi keempat

4.33 Mengisi Ruang Kosong

Ada barang-barang. Ada 3 ruang kosong. Barang itu hendak dimasukkan ke dalam ruangan. Volume ketiga ruangan sama. Barang dimasukkan satu persatu dengan cara berurutan. Ambil satu barang, masukkan ke ruangan. Ambil barang berikutnya, masukkan lagi ke dalam ruangan. Demikian seterusnya sampai barang terakhir.

Dari 3 ruangan itu, ke mana sebuah barang dimasukkan? Masukkanlah ke ruangan yang mempunyai sisa kapasitas terbesar. Jika ada lebih dari satu ruangan memiliki sisa kapasitas terbesar, pilih saja dengan bebas. Bagaimana kalau sisa kapasitas ruangan tidak mencukupi untuk sebuah barang? Barang tidak dimasukkan, ambil lagi barang berikutnya. Lakukan sampai barang terakhir, sehingga akan ada barang tersisa jika memang kapasitas ketiga ruangan sudah tidak ada lagi yang bisa diisi barang. Kita akan mencari berapa banyak barang tersisa dalam proses mengisi ketiga ruangan ini. Contohnya seperti berikut.

```
Antrian barang yang akan masuk   1 5 3 2 2 5 1 4 2 7 3 9 2
Proses memasukkan barang
-----------------------------------      Kapasitas ketiga
Barang    Masuk ke    Sisa Kapasitas     ruangan sama
                      A     B     C      A = 10
-----------------------------------      B = 10
1         A           9    10    10      C = 10
5         B           9     5    10
3         C           9     5     7
2         A           7     5     7
2         A           5     5     7
5         C           5     5     2
1         A           4     5     2
4         B           4     1     2
2         A           2     1     2
7         -           2     1     2
3         -           2     1     2
9         -           2     1     2
2         A           0     1     2
-----------------------------------
```

Jumlah barang yang tersisa adalah 3.

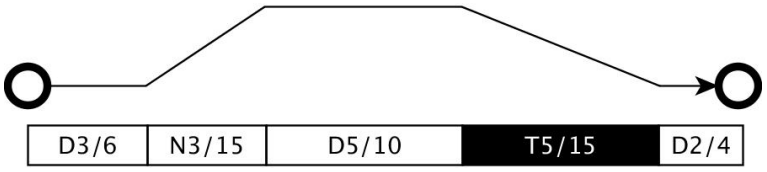
Data soal. Setiap baris berisi sejumlah angka. Setiap angka dipisahkan oleh satu spasi. Angka pertama adalah kapasitas setiap ruangan. Seperti deskripsi soal di atas, ada 3 ruangan yang akan diisi. Angka selanjutnya adalah antrian barang yang akan dimasukkan ke dalam ruangan. Hasil dari program adalah jumlah barang yang tersisa yang tidak masuk ke dalam ruangan.

```
10 2 2 2 4                          h#1: 0
5 10 10 10 10 10 10 20 20           h#2: 8
20 10 20 30 40 50 60                h#3: 4
```

4.34 Modal Minimal

Soal ini tentang mobil yang bisa mengumpulkan energi jika melewati jalan yang menurun. Untuk melewati jalan datar dan menanjak, mobil membutuhkan energi dalam jumlah tertentu. Kita buat contoh, untuk jalan datar, perlu 2 satuan energi per kilometer. Jalan menanjak, 5 satuan energi per kilometer. Sedangkan pada jalan yang menurun, mobil bisa mengumpulkan 3 satuan energi per kilometer.

Ada sebuah jarak yang akan ditempuh mobil ini. Disajikan dalam bentuk jarak yang datar, menanjak, dan menurun. Kode untuk jalan datar adalah D, jalan menanjak dengan kode N, dan jalan menurun dengan kode T. Misalnya ditulis D5, artinya jalan datar sepanjang 5 kilometer.



Gambar di halaman sebelumnya adalah contoh perjalanan dengan kode `D3 N3 D5 T5 D2`. Pada bagian bawah gambar ada kode jarak dan energi yang digunakan, atau energi yang dikumpulkan pada jalan menurun. Dalam contoh ini, agar mobil sampai ke tujuan, di awal harus ada minimal 31 satuan energi yang dimilikinya. Tentang kapasitas menyimpan energi, mobil ini dapat menampung energi dalam jumlah tak terbatas. Ini model penyimpanan yang diidamkan banyak orang.

Data soal. Setiap baris berisi angka penggunaan dan pengumpulan energi, serta data jarak yang akan ditempuh mobil. Setiap angka dipisahkan oleh satu spasi. Tiga angka pertama terkait dengan konsumsi dan pengumpulan energi per kilometer. Angka pertama konsumsi energi per kilometer pada jalan datar. Angka kedua konsumsi energi per kilometer pada jalan menanjak. Angka ketiga adalah pengumpulan energi per kilometer pada jalan menurun. Sedangkan angka selanjutnya adalah jarak yang akan ditempuh mobil. Hasil dari program adalah jumlah energi minimal pada awal saat mobil akan berangkat, agar mobil dapat mencapai tujuan akhir.

```
2 2 2 1D 10N 10T                    h#1: 22
1 1 4 5T                            h#2: 0
1 5 3 20D                           h#3: 20
```

4.35 Mengikuti Dua Robot

Ada 2 robot yang berjalan dalam area dua dimensi. Jalur yang akan ditempuh robot ini ditulis dalam kode tertentu. Ada 4 huruf yang digunakan, yaitu U, S, T, dan B. Kode U artinya bergerak satu langkah ke Utara. Kode S, bergerak satu langkah ke Selatan. Kode T bergerak satu langkah ke Timur, dan kode B artinya bergerak satu langkah ke Barat. Kedua robot awalnya berada pada posisi yang sama. Masing-masing robot punya jalur sendiri. Kita akan mencari ada berapa titik yang berhimpitan pada kedua jalur robot tersebut. Kita lihat contohnya pada diagram berikut.

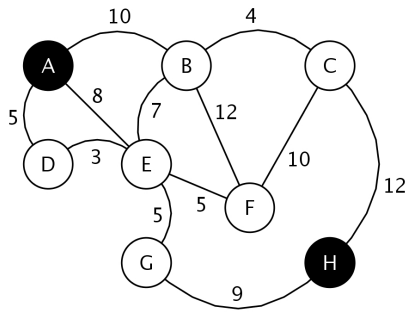
```
Jalur a : UUTTTSSSSTTUUU
Jalur b : TSTTSSBBBBB
. . . . . . . . . . . . . .
. . . . . . . . . . . . . .
. . . . . a a a a . . . . .
. . . . . a . . a . a . . .
. . . . . 0 b . a . a . . .
. . . . . . b b X . a . . .
. . . . . . . . X a a . . .
. . . b b b b b b . . . . .
. . . . . . . . . . . . . .
. . . . . . . . . . . . . .
```

Jalur yang ditempuh robot pertama ditandai dengan huruf `a` sedangkan jalur robot kedua ditandai dengan huruf `b`. Posisi awal diberi tanda dengan huruf `0`. Tanda `X` adalah titik dimana kedua robot mempunyai jalur yang sama. Dari contoh di atas, kedua robot mempunyai jalur yang sama pada 3 titik, yaitu 1 di posisi awal, dan 2 buah pada jalur yang dilewati.

Data soal. Setiap baris berisi kode jalur kedua robot. Data jalur pertama dan kedua dipisahkan oleh satu spasi. Kode jalur terdiri dari huruf U, S, T, dan B. Panjang jalur minimal adalah 1 huruf dan maksimal 1000 huruf. Hasil dari program adalah jumlah titik yang sama antara jalur 1 dan jalur 2. Minimal ada satu titik yang sama, karena kedua robot memulai dari posisi awal yang sama.

```
UUU TTT                                 h#1: 1
BBUU BBUU                               h#2: 5
TTSSSSS SSTTTTT                         h#3: 2
```

4.36 Jarak Dua Kota



Ada dua kota. Ada jalan yang menghubungkannya. Ada yang langsung, ada juga yang melewati kota lainnya. Jarak tempuhnya mungkin ada yang sama, tentu saja bisa berbeda. Dengan pengetahuan jalur yang ada serta berapa jauhnya, kita akan cari lintasan terpendek diantara kedua kota. Diagram di samping ini bisa memperjelas persoalan kita. Dari kota A menuju kota H, berapakah jarak terpendek?

```
Jalur yang bisa dilewati dari A menuju H
-------------------------------------------------
A B C H       ->  10 +  4 + 12            = 26 Km
A B F E G H   ->  10 + 12 +  5 +  5 +  9   = 41 Km
A D E G H     ->   5 +  3 +  5 +  9        = 22 Km
A D E F C H   ->   5 +  3 +  5 + 10 + 12   = 35 Km
A E G H       ->   8 +  5 +  9            = 22 Km
A E F C H     ->   8 +  5 + 10 + 12        = 35 Km
-------------------------------------------------
```

Jarak terpendek antara kota A dengan kota H adalah 22 Km. Kebetulan dalam diagram di atas bisa melalui dua jalur. Ada beberapa jalur lain yang tidak dicantumkan dalam daftar di atas. Jalur tersebut mempunyai jarak tempuh lebih besar dari 22 Km.

Data soal. Setiap baris berisi nama kota yang akan dicari jarak terpendeknya. Jumlah kota minimal 2, dan maksimal 10 kota. Nama kota menggunakan satu huruf yaitu A-J. Misalnya akan dicari jarak terpendek antara kota A dengan D, maka bagian pertama dalam baris soal berisi AD. Setelah itu dipisahkan satu spasi. Dilanjutkan dengan daftar jarak kota yang ada. Bentuknya adalah dua huruf merupakan nama untuk dua kota, kemudian diikuti angka yang menunjukkan jarak antara dua kota tersebut. Misalnya BC15, artinya adalah jarak antara kota B dengan kota C adalah 15 Km. Hasil dari program adalah jarak terpendek yang bisa ditemukan antara dua kota tersebut.

```
AC AB10 AC17 BC2                          h#1: 12
AE AB2 AC5 AD7 BC10 BD13 CD20 AE25        h#2: 25
AB AB7 BC1 CD2                            h#3: 7
```

4.37 Air Mengalir Sampai Jauh

Air secara alami mengalir dari tempat tinggi ke tempat yang rendah. Seperti sungai mengalir dari gunung ke lautan. Dengan menampilkan tinggi permukaan melalui matriks dua dimensi, kita akan mencari apakah air bisa mengalir sampai ke tujuannya. Ada syarat yang kita buat sederhana. Air bermula dari titik kiri atas. Kiri atas pada sebuah matriks adalah baris pertama kolom pertama. Tujuannya boleh ke sisi paling kanan atau sisi paling bawah. Sisi kanan artinya kolom paling kanan, sisi paling bawah adalah baris paling bawah dalam sebuah matriks.

Angka yang ada dalam matriks menunjukkan tinggi sebuah permukaan. Air hanya bisa mengalir jika permukaan yang ditujunya sama tinggi atau lebih rendah. Jika lebih tinggi, tentu saja air tak mengalir ke sana. Sedangkan arah yang boleh dilewati dari sebuah titik adalah ke atas, ke bawah, ke kiri, atau ke kanan. Jika diberikan sebuah matriks ketinggian permukaan ini, maka kita akan mencari apakah air dari posisi awal di kiri atas bisa sampai ke tujuannya. Contohnya sebagai berikut.

```
9* 7= 7= 7= 7= 7= 7= 7= 7= 9  9  7  2
5= 5= 5= 5= 8  8  8  8  4= 9  9  6  2
6  8  5= 8  9  9  9  7  4= 4= 4= 6  6
3  8  4= 4= 4= 4= 4= 7  9  9  9  3  2
3  8  7  7  6  6  3= 7  9  9  9  7  3
3  1  1  7  7  7  3= 7  9  9  9  7  2
4  2  7  1= 2= 2= 3= 7  9  9  9  7  3
5  7  7  5  5  8  2= 7  9  9  9  7  3
5  8  8  8  8  8  2= 8  8  8  8  7  3
```

Posisi awal diberi tanda bintang (*). Jalur air mengalir diberi tanda samadengan (=) di samping angka. Dalam contoh ini air dapat mencapai sisi paling bawah. Pada jalur yang satu lagi ke arah kanan, air tidak sampai ke kolom paling kanan. Tetapi kita hanya perlu mengetahui apakah air bisa sampai ke salah satu tujuannya, yaitu sisi paling kanan atau sisi paling bawah. Dalam hal ini, air berhasil mencapai tujuannya.

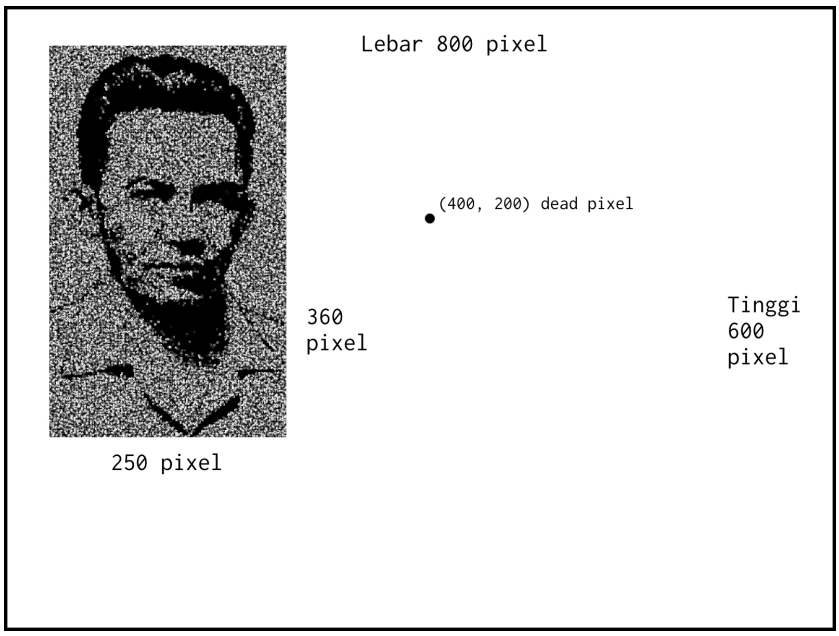
Data soal. Setiap baris berisi data ketinggian permukaan. Setiap angka dipisahkan oleh satu spasi. Matriks dalam bentuk bujursangkar n x n, sehingga jumlah angka dalam satu baris merupakan pangkat dua dari panjang sisinya. Susunlah angka tersebut menjadi matriks n x n, kemudian cari apakah air dapat mengalir sampai ke salah satu sisi kanan atau sisi bawah dari matriks tersebut. Ukuran matriks minimal adalah 2 x 2, dan maksimal 100 x 100. Hasil dari program berupa kata YA jika air berhasil mencapai tujuan, dan kata TIDAK jika air tidak bisa mencapai tujuannya.

```
2 3 4 3 1 1 4 4 4              h#1: TIDAK
5 5 5 4 4 4 1 1 1              h#2: YA
9 9 9 9                        h#3: YA
```

4.38 Dead Pixel

Gambar yang tampil pada layar monitor komputer terdiri dari banyak titik. Dengan resolusi sebesar 1024 x 768, maka ada 786432 titik yang disusun dalam berbagai warna dan kecerahan. Titik sebagai elemen gambar pada monitor disebut dengan *pixel*. Adakalanya beberapa titik pada layar itu rusak. Rusak dalam artian tidak lagi menyala dan menampilkan cahaya (*dead pixel*).

Sebuah gambar digital juga disusun oleh banyak titik. Tergantung panjang dan lebar gambar, maka jumlah titik yang menyusun sebuah gambar digital adalah hasil perkalian panjang dan lebar gambar tersebut. Sebuah gambar dengan lebar 100 *pixel* dan tinggi 200 *pixel* bisa ditampilkan pada layar yang berdimensi sama atau lebih besar dari gambar ini.



Sekarang kita punya kasus. Salah satu titik pada layar mengalami kerusakan. Posisinya bisa dimana saja. Untuk contoh, layar dengan lebar 800 *pixel* dan tinggi 600 *pixel* mengalami kerusakan satu buah di *pixel* di koordinat 400, 200. Angka 400 artinya kolom ke 400 dari sebelah kiri. Angka 200 artinya baris ke 200 dari atas. Jika kita punya sebuah gambar dengan ukuran 250 x 360 *pixel*, apakah gambar ini bisa ditampilkan utuh pada layar yang mengalami kerusakan satu *pixel* ini? Tampil utuh artinya semua *pixel* pada gambar dapat ditampilkan.

Melihat diagram pada halaman sebelumnya, kita bisa menampilkan dengan utuh gambar tersebut. Tetapi jika gambar kita berukuran 500 x 500 *pixel*, tidak bisa ditampilkan dengan utuh, karena pasti akan ada satu *pixel* mati terdapat dalam gambar tersebut.

Data soal. Setiap baris terdapat 6 angka. Setiap angka dipisahkan oleh satu spasi. Angka pertama dan kedua adalah ukuran monitor, yaitu lebar dan tinggi. Angka ketiga dan keempat adalah ukuran gambar yang akan ditampilkan pada monitor, yaitu lebar dan tinggi. Angka kelima dan keenam adalah posisi *dead pixel*, yaitu kolom dan baris. Kolom dihitung mulai dari kiri. Kolom paling kiri bernomor 1. Baris dihitung dari atas. Baris paling atas diberi nomor 1. Hasil dari program adalah kata YA jika gambar bisa ditampilkan dengan utuh tanpa *dead pixel*. Kata TIDAK jika gambar tidak bisa ditampilkan dengan utuh. Panjang sisi sebuah layar minimal 10, maksimal 10^4. Ukuran panjang sisi gambar minimal 10, maksimal 10^4. Setiap layar selalu ada satu buah *dead pixel*.

```
100 100 100 100 5 5                 h#1: TIDAK
800 600 400 400 120 110             h#2: YA
1000 1000 200 200 500 500           h#3: YA
```

4.39 Angka yang Disebutkan

Program komputer yang berhubungan dengan keuangan biasanya mempunyai fitur untuk menyebutkan sebuah bilangan. Contoh yang lazim adalah kuitansi. Sebuah kuitansi bukti pembayaran berisi angka 230.000, disertai dengan penyebutan dalam bahasa lokal. Dalam bahasa Indonesia penyebutannya adalah **dua ratus tiga puluh ribu**. Bagaimana kalau kita lakukan sebaliknya? Sebuah kalimat penyebutan bilangan akan dikonversi menjadi bilangan. Misalnya kalimat **enam ratus tiga puluh satu** menjadi 631.

Data soal. Setiap baris berisi kalimat yang menyebutkan sebuah bilangan. Semua kata disambung menjadi satu, tanpa ada spasi. Angka yang disebutkan minimal adalah satu dan maksimal sembilan ratus sembilan puluh sembilan juta sembilan ratus sembilan puluh sembilan ribu sembilan ratus sembilan puluh sembilan (999.999.999). Hasil dari program adalah bilangan dari kalimat tersebut.

```
seratusribuseratus                  h#1: 100100
duapuluhtujuh                       h#2: 27
empat                               h#3: 4
```

4.40 Pseudocode

Pseudocode[4] bukanlah kode program yang memiliki kelengkapan untuk dijalankan. Tujuannya adalah untuk menggambarkan sebuah algoritma dalam gaya bahasa pemrograman. Saya mengambil sebuah *pseudocode* dari buku algoritma[5].

```
int hash = 0;
for (int i = 0; i < s.length(); i++)
    hash = (R * hash + s.charAt(i)) % M;
```

Algoritma ini digunakan untuk membuat *hash*[6] dari sebuah string. R dan M dalam baris ketiga merupakan konstanta. Kita akan ganti R dengan angka 31, dan M dengan angka 65536. Tanda persen (%) merupakan operator modulo. Berikut ini penggambaran untuk algoritma di atas. Kata yang akan dihitung nilai *hash*nya adalah kata `bulan`. Untuk keperluan ini kita memerlukan tabel nilai untuk setiap huruf a-z.

```
Tabel nilai huruf ----------------------------------------
a =  97  b =  98  c =  99  d = 100  e = 101  f = 102
g = 103  h = 104  i = 105  j = 106  k = 107  l = 108
m = 109  n = 110  o = 111  p = 112  q = 113  r = 114
s = 115  t = 116  u = 117  v = 118  w = 119  x = 120
y = 121  z = 122
------------------------------------------------------------
          hash = 0
huruf b : hash = (31 *     0 +  98) % 65536 -> hash = 98
huruf u : hash = (31 *    98 + 117) % 65536 -> hash = 3155
huruf l : hash = (31 *  3155 + 108) % 65536 -> hash = 32377
huruf a : hash = (31 * 32377 +  97) % 65536 -> hash = 20744
huruf n : hash = (31 * 20744 + 110) % 65536 -> hash = 53350
------------------------------------------------------------
Nilai hash dari kata bulan adalah : 53350
```

[4] **pseudocode** *computing* A description of a computer programming algorithm that uses the structural conventions of programming languages but omits detailed subroutines or language-specific syntax. — `http://en.wiktionary.org/wiki/pseudocode`

[5] **Algorithms**, karangan Robert Sedgewick dan Kevin Wayne. Fungsi ini ada pada bab Searching bagian Hash Tables

[6] A **hash function** is any algorithm that maps data of variable length to data of a fixed length. The values returned by a hash function are called hash values, hash codes, hash sums, checksums or simply hashes. — `http://en.wikipedia.org/wiki/Hash_function`

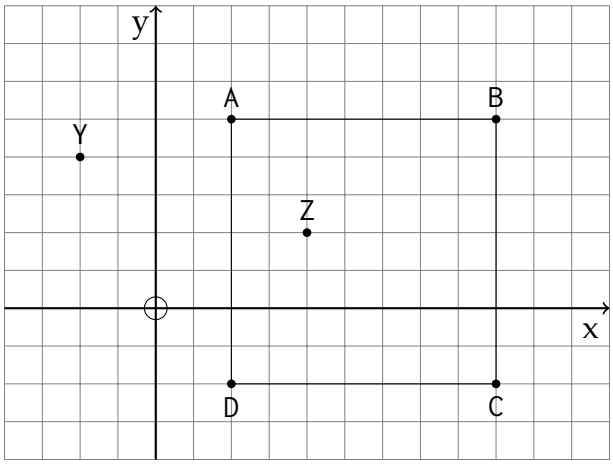
Data soal. Setiap baris berisi sebuah kata. Kata menggunakan huruf a-z. Hitunglah nilai *hash* dari kata tersebut menggunakan cara yang dijelaskan dalam soal ini.

```
kelinci                                  h#1: 35229
merpati                                  h#2: 36096
tupai                                    h#3: 34423
```

4.41 Titik Dalam Segiempat

Sebuah segiempat dalam koordinat 2 dimensi dapat dinyatakan dengan 4 titik koordinat. Empat buah titik ini adalah titik sudut dari segiempat tersebut. Ada titik lain, sebut saja Z. Kita akan menentukan apakah titik Z ini berada dalam segiempat, atau di luar segiempat. Contohnya dibuat dalam diagram berikut ini.

```
Empat titik sudut        Titik Z
A = (2,5)                yang akan ditentukan
B = (9,5)                posisinya, di dalam
C = (9,-2)               atau di luar segiempat
D = (2,-2)               Z = (4,2)
```



Menggunakan grafik di atas, dengan mudah bisa diketahui bahwa titik Z berada di dalam segiempat. Sedangkan titik Y yang berada di koordinat (-2,4) berada di luar segiempat. Bagaimana halnya dengan titik A, B, C, dan D? Titik-titik ini berada di dalam segiempat.

Data soal. Setiap baris berisi data 5 buah titik dalam koordinat dua dimensi. Setiap data titik dipisahkan oleh satu spasi. Titik disajikan dalam dua angka (x,y). Titik pertama adalah titik Z yang akan ditentukan apakah berada di dalam atau di luar segiempat. Empat titik berikutnya adalah titik sudut dari segiempat. Hasil dari program adalah kaya YA jika titik Z berada dalam segiempat, dan kata TIDAK, apabila titik Z tidak berada dalam segiempat.

```
1,1 2,2 4,2 4,4 2,4                    h#1: TIDAK
0,0 0,0 5,0 5,8 0,8                    h#2: YA
-1,-1 0,0 0,-9 -9,-9 -9,0              h#3: YA
```

4.42 Memutar Kalimat

Sebuah pesan dapat dienkripsi, sehingga orang yang tidak mengetahui cara membukanya tidak dapat mengetahui isi pesan. Merahasiakan pesan adalah usaha yang sudah dilakukan sejak lama. Salah satu metode kuno yang sering disebutkan adalah *caesar cipher*, yang digunakan oleh kaisar Julius Caesar. Metode ini sederhana saja. Satu huruf dipetakan ke huruf lainnya, bisa dengan cara menggeser. Misalnya A jadi B, B jadi C, seterusnya, sampai pada Z jadi A. Tapi jauh sebelum itu ada literatur yang mengatakan bahwa Alkindi mempunyai pengetahuan yang cukup mahir dalam hal kriptografi pada zamannya.

Soal ini tidak jauh terkait dengan cerita di atas. Kita akan melakukan enkripsi terhadap kalimat dengan cara memutarnya. Memutar, dalam artian yang sesungguhnya, kalimat yang terdiri dari rangkaian huruf ini, akan diputar dalam jalur spiral. Dimulai dari titik tengah, menuju keluar. Titik tengah adalah huruf pertama dalam kalimat. Ketika menyusun dalam bentuk spiral ini, semua spasi tidak disertakan. Kalimat yang akan diputar adalah: *langit langit akhlak rubuh di atas negeriku*.

```
-------------------------------------------------------
                        l    la   la   lan  lang  lang
l  la  la  la   la  tla  tla  tla  tla  tla  tla   tlai
        n  gn  ign  ign  ign  ign  ign  ign  ign   ign
-------------------------------------------------------
Hasil akhir kalimat                  buhdia         1
setelah diputar                      ulangt         2
tampak seperti                       rtlaia         3
di samping ini                       kignts         4
                                     alhkan         5
                                    ukirege         6
-------------------------------------------------------
```

Proses memutar kalimat digambarkan dengan penambahan satu demi satu huruf. Berapa bariskah hasilnya setelah diputar? Dari diagram dapat dilihat hasil akhir sebanyak 6 baris.

Data soal. Setiap baris berisi kalimat yang akan diputar dengan cara ini. Satu kata terdiri dari huruf a-z. Pemisah antar kata adalah satu buah spasi. Panjang kalimat maksimal adalah 1000 huruf, termasuk spasi. Panjang minimal adalah 1 huruf. Program mencari berapa baris hasil akhir setelah diputar.

```
sang petualang                          h#1: 4
paman doblang                           h#2: 3
air mata                                h#3: 3
```

4.43 Ketikan yang Rusak

Komputer menggunakan *keyboard* yang menyerupai tombol di mesin tik. Dalam kondisi baik, satu kali menekan tombol di papan ketik ini akan menghasilkan satu buah huruf. Dalam keadaan tertentu, karena mengalami kerusakan, bisa saja sekali menekan tombol huruf yang dihasilkan lebih dari satu. Bisa dua, tiga atau lebih huruf yang sama. Sebelumnya kita punya asumsi yang sederhana, bahwa tidak ada dalam kalimat sebuah huruf berada secara berdampingan. Nah, soalnya adalah, hasil ketikan yang memunculkan banyak huruf bersamaan ini perlu diperbaiki. Perbaikannya adalah dengan membuang huruf yang ganda, dijadikan satu huruf saja. Contohnya sebagai berikut.

```
------------------------------------------------------------
appiiiiiii revvolusi haruskkkahhhhhhh ppaaddaamm digantikan
fffigur    yaaaaaaaang ttaakkkkk      paaaasssssstii

api revolusi haruskah padam digantikan figur yang tak pasti
------------------------------------------------------------
```

Data soal. Setiap baris berisi satu buah kalimat. Panjang maksimal kalimat adalah 1000 huruf, minimal 1 huruf. Kalimat ini berisi huruf yang salah ketik, yaitu huruf tertulis ganda. Program membersihkan huruf yang salah ini, dan menghasilkan kalimat yang sudah dikoreksi.

```
noccccccttturnooooooooooo          h#1: nocturno
kesakssssssssian                   h#2: kesaksian
aaaaaaaaaaaaaaaaaaa                h#3: a
```

4.44 Berapa Baris?

Fungsi yang memberikan kemudahan dalam sebuah aplikasi penyunting teks adalah fungsi memotong kalimat pada batas paling kanan area penulisan. Jika kalimat yang kita tulis sampai di batas tersebut, maka secara otomatis akan berpindah ke baris berikutnya. Fungsi ini tentu saja memerlukan kalkulasi. Perhitungannya bisa sederhana, bisa juga kompleks. Apalagi jika melibatkan pemotongan di tengah kata, dengan mengikuti kaidah sebuah bahasa dalam memenggal kata. Penyusunan algoritma dan kode komputasinya bisa jadi rumit.

Untuk mencoba seberapa menariknya problem ini diselesaikan, kita akan menggunakan beberapa persyaratan yang ringan. Kita akan menghitung berapa baris yang akan ditempati sebuah paragraf yangf terdiri dari beberapa kalimat. Satu buah kalimat terdiri dari beberapa kata yang diakhiri oleh titik. Tanda titik berada tepat setelah kata terakhir dalam kalimat tersebut, tidak dipisahkan oleh spasi. Sedangkan pemisah antar kata adalah satu buah spasi. Lebar yang bisa ditempati dalam satu baris adalah sebanyak 40 karakter. Karakter yang terlibat dalam sebuah paragraf adalah A-Za-z, titik, koma, dan spasi. Pemotongan di akhir baris hanya boleh dilakukan pada spasi. Tidak boleh memotong di tengah kata. Satu baris tidak boleh melebihi batas maksimal jumlah karakter yang ditentukan. Sebaliknya tentu saja boleh apabila kurang dari batas maksimal. Berikut ini contohnya. [7]

```
Sekilas, kembali ia menatap ke mata putrinya. Tetapi kabut
itu, tiga dunia itu, bagai deras menolak, mendorong ia
balik ke alam nyata, segala yang terbentang di hadapannya.
Undukan undukan itu, pepohonan yang teratur, vila vila,
Bukit Burai yang berubah. Terbayang pula ladang mereka,
ladang ladang penduduk sekitar dan kaumnya, yang telah
jauh pindah ke lembah. Orang bunian itu, masih adakah?
Sangat ingin ia menjawab: Tidak. Tetapi, lirih, mulutnya
berkata, "Masih".
```

Diatur satu baris maksimal 40 karakter. Hasilnya 13 baris.

[7] Paragraf dalam contoh ini dikutip dari cerpen "Orang Bunian" karya Gus tf Sakai

```
---------1---------2---------3---------4
Sekilas, kembali ia menatap ke mata                    1
putrinya. Tetapi kabut itu, tiga dunia                 2
itu, bagai deras menolak, mendorong ia                 3
balik ke alam nyata, segala yang                       4
terbentang di hadapannya.                              5
Undukan undukan itu, pepohonan yang                    6
teratur, vila vila, Bukit Burai yang                   7
berubah. Terbayang pula ladang mereka,                 8
ladang ladang penduduk sekitar dan                     9
kaumnya, yang telah jauh pindah ke                    10
lembah. Orang bunian itu, masih adakah?               11
Sangat ingin ia menjawab: Tidak. Tetapi,              12
lirih, mulutnya berkata, "Masih".                     13
---------1---------2---------3---------4
```

Data soal. Setiap baris berisi angka yang menunjukan lebar maksimal satu baris, kemudian diikuti oleh teks untuk satu buah paragraf. Pemisah antara angka lebar baris dengan teks paragraf adalah satu buah spasi. Hasil program adalah jumlah baris yang diperlukan untuk satu paragraf tersebut. Panjang minimal satu paragraf adalah 1 karakter, maksimal 1000 karakter. Contoh berikut hanyalah untuk ilustrasi karena keterbatasan ruang.

```
55 Mereka belum juga...              h#1: 2
30 Siapakah yang menerbitkan n...    h#2: 27
60 Jangan berhenti. Disana tida...   h#3: 4
```

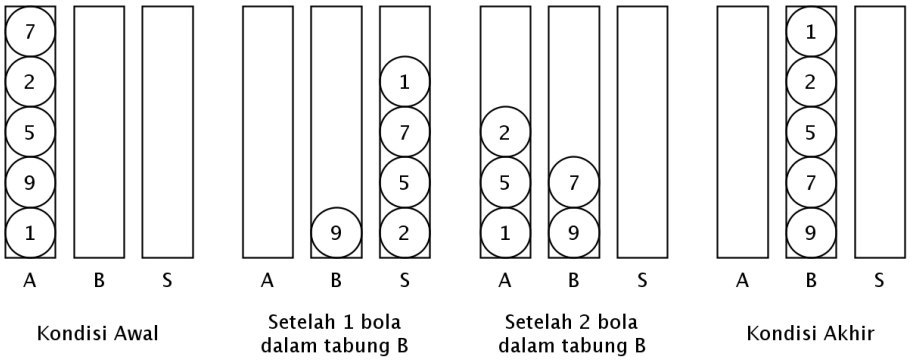
4.45 Tiga Tabung

Ada tiga tabung. Bayangkanlah tabung untuk menyimpan bola tenis. Ketiganya akan digunakan untuk menyusun bola. Setiap bola mempunyai nomor. Antara satu bola dengan bola lainnya tidak ada nomor yang sama. Satu tabung pada awalnya berisi sejumlah bola. Dengan memanfaatkan tiga tabung ini kita akan menyusun bola secara berurutan sesuai nomornya.

Tabung pertama kita sebut saja tabung A. Tabung kedua namanya B. Tabung ketiga bernama S. Tabung yang awalnya berisi adalah tabung A. Tabung B akan menjadi tujuan akhir, dimana semua bola dari tabung A akan masuk ke tabung ini dalam keadaan terurut sesuai nomor yang ada pada masing-masing bola. Bola dengan nomor terbesar ada di dasar tabung, dan nomor terkecil pada posisi paling atas. Untuk apakah tabung S. Tabung ini dipakai sebagai penampung sementara jika diperlukan. Selain tiga tabung, tangan kanan dan kiri juga menjadi bagian dalam proses ini.

Bagaimana cara kerjanya? Dalam satu waktu, kita hanya bisa mengambil satu bola dari setiap tabung. Menggunakan tangan kanan, diambil bola teratas dari tabung A. Dari tangan kanan, bola ini diletakkan ke tangan kiri. Tangan kanan mengambil bola berikutnya dari tabung A. Bola ini dibandingkan dengan bola yang ada di tangan kiri. Jika bola yang di tangan kanan nomornya lebih besar dari bola yang ada di tangan kiri, maka bola yang di tangan kiri dimasukkan ke tabung S. Bola yang ada di tangan kanan ini diletakkan ke tangan kiri. Ambil lagi bola dari tabung A, bandingkan lagi dengan yang ada di tangan kiri. Artinya, setiap kali bola diambil dari dalam tabung, dibandingkan dengan bola yang ada di tangan kiri. Bola yang angkanya lebih besar harus selalu diletakkan ke tangan kiri.

Lanjutkan proses ini sampai bola di tabung A habis. Ketika semua bola sudah habis dari tabung A, maka ada satu bola di tangan kiri dan sisanya ada di tabung S. Bola di tangan kiri dimasukkan ke dalam tabung B.



Sampai di sini, tabung B berisi satu bola, yaitu bola dengan nomor terbesar. Kita ulangi lagi langkah yang sama. Sekarang bola berasal dari tabung S. Tangan kiri digu-

nakan untuk meletakkan bola yang bernomor lebih besar. Tangan kanan dipakai untuk mengambil bola dari tabung. Tabung A dan S menjadi tempat penampungan sementara. Pada akhir proses kedua, satu bola ada di tangan kiri, sisanya dalam tabung A. Bola di tangan kiri masuk ke tabung B. Sudah ada dua bola di tabung B.

Langkah ini terus dilakukan sampai semua bola masuk ke tabung B. Posisi bola di tabung B adalah: bola dengan nomor terbesar berada di bawah, bola dengan nomor terkecil paling atas. Nomor bola sudah terurut.

Apa yang hendak jadi soal? Hitunglah berapa kali bola di tangan kiri masuk ke tabung S dan tabung A. Berikut ini adalah uraian langkah sesuai dengan gambar tabung seperti terlihat di atas.

```
---------   ---------   ---------   ---------   ---------
A a i S B   S a i A B   A a i S B   S a i A B   A a i S B
---------   ---------   ---------   ---------   ---------
7 7 7       1 1 1       2 2 2       1 1 1       1 1 1
2 2   2     7 7         5 5         2 2             1   1
5 5   5         1=1         2=2         1=1     ---------
9 9             7 7         5 5         2 2
    7=7     5 5   5     1 1   1           2   2
  9 9       2 2   2         5   5   ---------
1 1   1         7   7   ---------
    9   9   ---------
---------               A,S,B:Tabung a,i:Tangan kanan/kiri
```

Perpindahan bola yang diberi tanda samadengan (=) adalah bola dari tangan kiri ke tabung A dan dari tangan kiri ke tabung S. Ada 4 kali langkah ini terjadi.

Data soal. Setiap baris berisi sejumlah angka. Setiap angka dipisahkan oleh satu spasi. Jumlah angka minimal 1, dan maksimal 1000. Posisi angka dalam sebuah baris adalah posisi awal di tabung A. Angka pertama berada paling atas, dan angka terakhir berada pada posisi paling bawah. Program mencari berapa kali tangan kiri memasukkan bola ke tabung A dan tabung S.

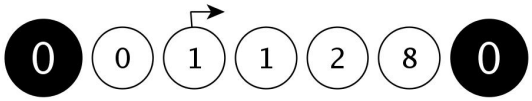
```
2 1                                             h#1: 0
5 4 1                                           h#2: 1
9 1                                             h#3: 0
```

4.46 Congklak Tanpa Lawan

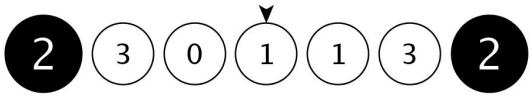
Permainan tradisional congklak dulu pernah jadi ajang untuk unjuk ketrampilan. Masihkah kanak-kanak memainkannya hari ini? Mungkin ada. Mereka mengenalnya melalui sentuhan di layar digital.

Congklak biasanya dimainkan berdua. Dengan tujuh lobang berisi kerang sebagai buah congklak. Di ujungnya ada lumbung besar untuk menabung keuntungan. Congklak yang akan kita libatkan tidak persis sama. Ada modifikasi. Lumbungnya tetap 2, ada di ujung kiri dan kanan. tetapi tidak ada lawan yang akan bertanding. Hanya ada satu baris lobang untuk diisi buah congklak.

Kita contohkan, congklak 5 lobang dengan 2 lumbung di ujungnya. Pada awal permainan, setiap lobang bisa ada isinya, bisa juga kosong. Isinya acak, tidak sama. Permainan dimulai dari lobang yang isinya paling sedikit. Jika ada yang sama, maka ambil yang berada di lobang paling kiri. Arah permainan dimulai ke kanan. Setelah mengisi lumbung paling kenan, balik ke arah kiri, sampai ke lumbung ujung kiri, lalu kembali ke kanan. Demikian seterusnya sampai buah congklak yang ditangan berhenti di lobang yang kosong. Ketika berhenti di lobang kosong, maka permainan selesai. Jika pengisian berhenti pada salah satu lumbung, maka permainan masih berlanjut. Dari lobang mana dilanjutkan? Dari lobang dengan buah congklak paling sedikit. Jika lobang dengan isi paling sedikit ada lebih dari satu, maka ambil yang paling kiri. Langkahnya kembali dimulai ke arah kanan. Pada akhir permainan dihitung berapa banyak buah congklak yang berhasil dikumpulkan dalam dua lumbung.



Kondisi Awal



Kondisi Akhir

Langkah pertama dimulai dari lobang yang berisi 1 buah kerang. Karena ada dua lobang yang berisi 1, maka diambil yang paling kiri. Gambar pertama menunjukkan kondisi awal. Permainan dimulai dari lobang dengan tanda panah ke kanan. Gambar kedua adalah akhir permainan. Langkah terakhir pada lobang yang diberi tanda panah. Kerang yang berhasil dikumpulkan pada kedua lumbung adalah 4.

Data soal. Setiap baris berisi angka yang dipisahkan oleh satu spasi. Setiap angka menunjukkan jumlah awal buah congklak yang ada di setiap lobang. Angka ini tidak termasuk angka untuk lumbung. Lumbung kiri dan kanan awalnya selalu kosong. Angka minimal adalah 0, dan maksimal 1000. Jumlah lobang minimal 1, maksimal 100. Hasil dari program adalah jumlah buah congklak yang berhasil dikumpulkan di akhir permainan.

```
0 0 0 0 0                                   h#1: 0
1 1 0 0 0 3 0 0 0                           h#2: 0
3                                           h#3: 3
```

4.47 Lantai Terakhir

Sebuah elevator biasanya berjalan satu arah sampai tidak ada permintaan ke lantai tertentu. Misalnya elevator sedang bergerak ke atas. Elevator akan naik terus sampai lantai tertinggi sesuai permintaan dari penumpang. Demikian juga ketika bergerak ke bawah, akan terus ke bawah sampai lantai paling rendah sesuai permintaan.

Kita membuat cara tertentu untuk pergerakan elevator. Aturannya adalah: elevator akan menuju lantai terdekat dari posisinya saat ini. Tentu saja lantai terdekat yang ada dalam permintaan penumpang. Misalnya ada penumpang naik di lantai 1 hendak ke lantai 5. Sampai di lantai 2 naik seorang penumpang yang hendak menuju lantai 1. Dalam situasi ini elevator tidak naik ke lantai 5, karena tujuan terdekat dari lantai 2 adalah lantai 1. Elevator akan turun ke lantai 1, baru kemudian naik ke lantai 5. Jika algoritma ini diterapkan, adakah yang bisa membayangkan wajah penumpang yang hendak naik ke lantai 5?

Baiklah kita lanjutkan aturannya. Jika pada sebuah lantai ada dua permintaan dengan jarak yang sama, maka elevator bergerak ke arah yang sama dengan gerakan sebelumnya. Setiap elevator berhenti untuk menurunkan atau menaikkan penumpang, maka elevator akan melihat kembali tujuan terdekatnya. Baru diambil keputusan, akan bergerak ke mana. Kondisi awal elevator selalu dimulai dari lantai 1. Berikut contoh pergerakan elevator, sekaligus kita akan menjawab, di lantai berapakah elevator terakhir kali berhenti.

Lantai	1	2	3	4	5	6	7
Permintaan	2	7	2	0	2	0	3

Tabel nomor lantai dan permintaan tujuan

Perjalanan Elevator: $1 \Rightarrow 2 \Rightarrow 3 \Rightarrow 2 \Rightarrow 5 \Rightarrow 7 \Rightarrow 3 \Rightarrow 2$

Pada awalnya semua orang yang ada di lantai tertentu meminta untuk menggunakan elevator. Pengatur elevator mengetahui ada sejumlah permintaan dan dari lantai berapa permintaan itu dilakukan. Permintaan pertama tentu saja dari lantai 1. Ketika penumpang lantai 1 naik, dia menekan angka 2. Artinya sekarang ada tambahan baru, yaitu permintaan ke lantai 2. Elevator menuju lantai dua karena memang itu tujuan terdekatnya. Setelah di lantai dua, ada penumpang naik, dengan permintaan baru, yaitu lantai 7. Permintaan terdekat saat di lantai 2 adalah lantai 3. Maka elevator bergerak ke lantai 3. Sampai di lantai 3, ada permintaan baru, yaitu ke lantai 2. Walaupun elevator sedang menuju ke atas, karena permintaan terdekat adalah lantai 2, maka elevator turun ke lantai 2. Setelah di lantai dua, elevator menuju tujuan berikutnya yang terdekat yaitu lantai 5. Di lantai 5 ada permintaan baru menuju lantai 2. selain itu ada juga permintaan ke lantai 7. Jarak terdekat adalah lantai 7, maka elevator naik ke lantai 7. Di lantai 7

ada permintaan baru menuju lantai 3. Elevator turun ke lantai 3, kemudian terakhir ke lantai 2.

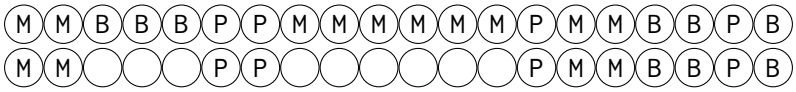
Data soal. Setiap baris berisi sejumlah angka. Setiap angka dipisahkan oleh satu spasi. Angka berisi nomor lantai yang diinginkan oleh pengguna. Angka minimal adalah 0, artinya tidak ada permintaan penggunaan elevator di lantai tersebut. Angka maksimal adalah angka yang sama dengan jumlah lantai. Angka pertama adalah angka permintaan dari lantai 1. Angka kedua adalah permintaan dari lantai 2, dan seterusnya. Pada keadaan awal elevator berada di lantai 1. Jumlah lantai maksimal adalah 100. Hasil dari program adalah di lantai berapa elevator terakhir kali berhenti.

```
0 1 0 0 0                                   h#1: 1
2 5 0 0 4                                   h#2: 4
0 0 0 0 0                                   h#3: 1
```

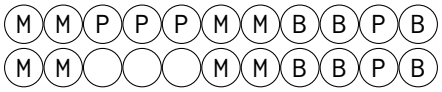
4.48 Kelereng Warna Warni

Permainan ini menggunakan kelereng. Ada 4 warna yang tersedia. Putih, Merah, Kuning, Biru. Kita singkat saja menjadi P, M, K, dan B. Kelereng disusun dalam satu baris yang panjang. Mulai dari pangkalnya, akan diperiksa susunan kelereng tersebut. Jika ada 3 atau lebih kelereng bersusun dengan warna sama, maka kelereng tersebut dikeluarkan. Jika ada 3 atau 4 yang sama, maka kita mendapat nilai 3. Jika warna yang sama 5 kelereng atau lebih, maka dapat nilai 5. Pemeriksaan ini berlanjut sampai ke ujung barisan.

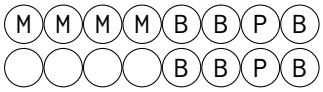
Setelah pemeriksaan selesai, nilai sudah dihitung, dan kelereng sudah dikeluarkan, akan ada ruang kosong dalam barisan tersebut. Langkah berikutnya semua kelereng dirapatkan, sehingga tidak ada ruang kosong antar kelereng.




```
Nilai : 3 + 5 = 8
```




```
Nilai : 3
```




```
Nilai : 0
```

Dimulai lagi dari pangkal barisan, proses memeriksa kelereng yang berwarna sama kembali dilakukan. Proses penilaian juga sama dengan cara sebelumnya. Langkah ini dilakukan terus sampai tidak ada lagi kelereng yang dikeluarkan dari barisan. Pada akhir permainan, berapakah nilai yang bisa dikumpulkan? Pada contoh ini ada 4 kali pemeriksaan dilakukan dengan total nilai 8 + 3 + 3 = 14. Pemeriksaan terakhir tidak ada nilai yang diperoleh, karena tidak ada lagi 3 atau lebih warna yang sama berdekatan.

Data soal. Setiap baris berisi sejumlah kelereng yang menggunakan kode warna. Jumlah minimal kelereng adalah 1 dan maksimal adalah 1000 kelereng. Hasil dari program adalah total nilai yang bisa didapatkan dalam permainan dengan aturan seperti di atas.

```
MPBMPBMPB                                h#1: 0
PPPMMMKKKBBB                             h#2: 12
MMKKMMMMMMMMMMM                          h#3: 5
```

4.49 Steganografi

Steganografi[8] adalah sebuah seni menyembunyikan informasi. Dalam teknologi informasi tentu saja cara ini bisa digunakan. Salah satunya adalah menyembunyikan pesan dalam sebuah gambar. Gambar yang akan digunakan berupa gambar digital dalam bentuk *greyscale*. Pada soal ini Setiap titik dalam gambar mempunyai nilai dari 0 sampai 63. Untuk menyimpan satu buah huruf, kita memerlukan 7 titik pada gambar. Atau sebaliknya, tujuh buah titik menjadi penyimpan untuk satu huruf. Proses ini dikerjakan dalam angka biner.

Langkah pertama adalah menyiapkan sebuah gambar minimal mempunyai titik sejumlah huruf yang akan disembunyikan. Kemudian setiap titik pada gambar diseragamkan pada bit terakhir jadi 0. Ini dilakukan dalam mode biner. Misalnya ada titik pada gambar dengan kode biner `1111101`, maka dijadikan `1111100`. Setelah semua bit terakhir untuk semua titik dijadikan nol, barulah akan disisipkan huruf ke dalam gambar ini.

Contoh sederhana, untuk satu huruf memerlukan tujuh titik. Huruf yang akan disembunyikan adalah **a**. Nilai desimalnya adalah 97, kode binernya adalah `1100001`. Setiap satu bit dari kode biner untuk huruf a ini akan disembunyikan pada satu titik. Karena ada 7 bit maka diperlukan 7 titik.

Nilai titik awal	Bit terakhir jadi 0	Kode biner yang akan disisipkan	Nilai setelah disisipi
111101	111100	1	111101
110010	110010	1	110011
110010	110010	0	110010
110101	110100	0	110100
111001	111000	0	111000
111101	111100	0	111100
111010	111010	1	111011

Huruf kedua dilanjutkan pada 7 titik berikutnya. Demikian seterusnya sampai semua huruf selesai disembunyikan. Diperlukan tabel nilai huruf dalam proses ini.

[8] **Steganography** hiding a secret message within a larger one in such a way that others can not discern the presence or contents of the hidden message. For example, a message might be hidden within an image by changing the least significant bits to be the message bits. — Free Online Dictionary of Computing

Huruf yang digunakan adalah a-z dengan kode desimal dari 97 sampai dengan 122.[9] Sedangkan spasi bernilai desimal 32.

Data soal. Setiap baris berisi sejumlah angka desimal. Angka ini merupakan nilai titik dalam sebuah gambar. Dalam titik-titik itu tersimpan sebuah pesan. Setiap angka dipisahkan oleh satu spasi. Minimal jumlah titik adalah 7, dan maksimal 7000 titik. Hasil dari program adalah menampilkan pesan yang tersimpan di dalamnya.

```
1 1 0 0 0 0 1                          h#1: a
31 31 20 20 20 21 8                    h#2: b
51 61 10 0 0 15 17                     h#3: c
```

[9]Lihat tabel nilai huruf a-z dalam soal Pseudocode

4.50 Tanpa Uang Kembalian

Uang digunakan untuk membayar sebuah barang yang dibeli. Ada beberapa pecahan mata uang. Misalnya 100, 500, 1000, dan seterusnya. Harga sebuah barang sangat beragam. Sedangkan uang hanya ada dalam pecahan terbatas, mungkin paling banyak dalam hitungan belasan. Dalam membayar sebuah barang sesuai dengan harganya, kita akan mencari apakah bisa menggunakan gabungan pecahan uang dengan pas. Tidak kurang, tidak lebih, tanpa kembalian.

Misalnya sebuah barang harganya adalah 750. Kita punya pecahan uang 500, 200, dan 25. Maka kita dapat membayarnya dengan gabungan 1 uang pecahan 500, 1 uang pecahan 200, dan 2 uang pecahan 25. Dengan pecahan yang sama, kita tidak bisa membeli barang yang berharga 430 dengan pembayaran yang pas, karena tidak ada gabungan dari pecahan uang yang menghasilkan harga yang sama dengan harga barang.

Data soal. Setiap baris berisi harga barang yang akan dibayar, dan beberapa pecahan uang yang dimiliki. Setiap angka dipisahkan oleh satu spasi. Angka pertama adalah harga barang. Angka berikutnya adalah pecahan uang. Tidak ada batasan berapa buah setiap pecahan digunakan. Hasil dari program adalah kata YA apabila bisa membayar dengan pas, dan kata TIDAK jika tidak ada gabungan pecahan uang yang bisa menghasilkan angka yang sama dengan harga barang.

```
4000 200 50 25                          h#1: YA
2200 1000                               h#2: TIDAK
5500 2000 200 100                       h#3: YA
```

4.51 Jam yang Bisa Dibalik

Jam digital menampilkan waktu menggunakan tampilan angka. Angka pada jam ini berbentuk *seven segment*. Jam ditampilkan menggunakan dua angka, begitu juga untuk menit ditampilkan dalam dua angka. Tidak ada angka untuk detik. Hanya jam dan menit. Antara jam dengan menit dipisahkan oleh tanda titik dua. Misalnya jam 10 lewat 12 menit akan tampil pada jam ini dalam bentuk `10:12`. Jam 2 lewat 3 menit dalam bentuk `02:03`.

Dengan tampilan *seven segment*, adakalanya jam ini bisa diletakkan terbalik, dan tetap menampilkan jam yang sama. Misalnya jam `12:21`, jika posisi jam ini dibalik, bagian atas di bawah, bawah menjadi atas, tampilannya akan tetap sama yaitu `12:21`. Mari kita lihat dalam gambar berikut.

Baliklah buku ini, maka angkanya tetap menampilkan `12:21`. Dalam gambar di atas disajikan angka 0 sampai 9 dalam tampilan *seven segment*.

Data soal. Setiap baris berisi rentang antara dua waktu. Waktu awal dan waktu akhir. Jika ditampilkan dalam jam digital ini, ada berapakah waktu yang bisa ditampilkan sama walaupun jam ini dibalik posisinya. Waktu minimal adalah `00:00`, dan maksimal `23:59`. Hasil dari program berupa angka yang menunjukkan jumlah jam yang bisa ditampilkan terbalik.

```
01:00 01:09                          h#1: 0
10:00 10:10                          h#2: 1
02:02 02:03                          h#3: 0
```

4.52 Tunjangan yang Dipotong

Gaji karyawan yang bekerja di sebuah perusahaan biasanya terdiri dari sejumlah elemen, seperti gaji pokok, lembur, bonus, tunjangan, dan lainnya. Karena sesuatu hal, gaji tersebut juga bisa dipotong. Mungkin karena cicilan, bisa juga karena tidak masuk kerja, atau potongan pajak. Banyak sekali variasinya. Makin beragam, makin repot menghitungnya, sementara kebutuhan hidup naik tanpa hitung-hitungan. Ah, kita ngelantur lagi.

Mari kembali ke persoalan tentang gaji dan tunjangan. Kita buat contoh, ada perusahaan yang memberi gaji kepada karyawan, misalnya sebesar 100.000 Rupiah. Karena ada aturan yang berlaku di sana bahwa gaji dipotong sebesar 5% untuk keperluan tertentu, maka gaji yang 100.000 tentunya akan dipotong sebesar 5000. Karyawan akhirnya akan menerima sebesar 100.000 - 5.000 = 95.000. Karyawan dan perusahaan sepakat bahwa karyawan tetap menerima 100.000. Juga potongan tidak berubah, tetap 5%.

Solusinya, perusahaan bersedia menambahkan ke gaji yang 100.000 tadi sebesar sekian rupiah, sehingga jika dipotong 5%, gaji yang diterima tetap 100.000. Nah persoalan sekarang adalah, berapa perusahaan menambah gaji karyawan tadi, sehingga ketika dikenai potongan 5%, gaji yang diterimanya adalah 100.000.

```
Dalam bentuk persamaan matematika menjadi :
(100000 + T) - ((100000 + T) x 0,05) = 100000
T adalah tambahan yang diberikan oleh perusahaan
A - 0,05A = 100000
    0,95A = 100000
        A = 105363,15789...
        T = A - 100000
        T = 5363,15789...
Dengan pembulatan ke atas, didapat angka sebesar 5364
```

Data soal. Setiap baris berisi gaji yang akan diterima karyawan, dan potongan dalam persen. Angka dipisahkan oleh satu spasi. Hasil dari program adalah besarnya tambahan yang diberikan perusahaan sesuai dengan contoh soal di atas. Hasilnya dalam bentuk bilangan bulat (*integer*). Jika diperlukan pembulatan, lakukan pembulatan ke atas. Angka gaji minimal 1000, maksimal 10^7. Potongan minimal 0%, dan maksimal 50%.

```
200000 10                                   h#1: 22223
850000 5                                    h#2: 94737
300000 0                                    h#3: 0
```

4.53 Antrian

Jika jumlah pelayan lebih sedikit dari jumlah yang akan dilayani, terjadilah antrian. Pihak yang dilayani perlu berbaik hati untuk menunggu gilirannya. Demikianlah budaya yang mulia. Sementara itu kita akan menyibukkan diri dengan perhitungan lagi. Soal budaya, endapkan sejenak.

Dalam antrian ada sekian orang yang akan dilayani. Bayangkan bahwa ini terjadi pada sebuah tempat cukur rambut. Tukang cukur ada 3 orang, sementara yang hendak bercukur ada 10. Dalam satu waktu hanya ada 3 orang yang bisa dilayani. Setiap orang juga memmerlukan waktu yang tak sama. Ada yang lama, ada yang lebih lama, tentu ada juga yang tak lama. Kita punya keperluan dengan angka. Berapa lamakah diperlukan waktu untuk mencukur semua orang ini. Waktunya dihitung sejak orang pertama mulai dicukur, sampai dengan orang terakhir selesai dilayani. Kita lihat dalam diagram berikut ini.

Urutan Antrian	1	2	3	4	5	6	7	8	9	10
Waktu Layanan	2	3	3	7	1	2	2	1	5	2

Tabel antrian

Tukang cukur 1	1(2)	4(7)		
Tukang cukur 2	2(3)	5(1)	7(2)	9(5)
Tukang cukur 3	3(3)	6(2)	8(1)	10(2)

Tabel layanan oleh tukang cukur
Angka dalam kurung adalah waktu pelayanan

Dalam diagram ada daftar orang yang akan dicukur sebanyak 10 orang. Setiap orang dilayani dalam waktu yang berbeda. Bagaimana model antrian ini? Jika ada tukang cukur yang kosong, maka orang yang ada dalam antrian terdepan akan dilayani oleh tukang cukur tersebut. Demikian seterusnya, setiap ada tukang cukur yang kosong, maka orang di antrian terdepan akan dilayani. Dari contoh di atas, dapat dilihat orang yang mana yang dilayani oleh tukang cukur yang mana. Waktu total yang diperlukan melayani semua orang adalah 11 menit.

Data soal. Setiap baris berisi sejumlah angka. Angka pertama adalah jumlah yang melayani. Minimal adalah 1, dan maksimal 100. Angka selanjutnya adalah data antrian. Setiap angka dipisahkan oleh satu spasi. Data untuk antrian berupa angka. Angka ini adalah waktu yang diperlukan untuk melayani satu orang. Jumlah angka adalah jumlah antrian. Minimal 1, dan maksimal 1000. Angka minimal untuk satu orang adalah 1, dan maksimal adalah 1000. Hasil program adalah waktu yang diperlukan untuk melayani antrian.

```
1 10 10 5 4                             h#1: 29
2 5 5 5 5 5 5                           h#2: 15
3 10 10                                 h#3: 10
```

4.54 Penyimpan Sementara

Cache adalah salah nama untuk *memory* yang dipakai dalam komputer. Berfungsi untuk menyimpan data, dengan tujuan agar proses bisa lebih cepat. Ukuran *cache* tidaklah besar, sehingga ada cara tertentu yang digunakan untuk menyimpan data di sana. Salah satu caranya adalah, data yang jarang dipakai akan turun prioritasnya. Jika semakin jarang, maka data ini bisa tidak lagi ada di *cache*. Sedangkan data yang sering dipakai, akan menempati prioritas utama.

Meskipun tidak terlalu tepat untuk kasus dalam soal ini, deskripsi di atas bisa memberi gambaran yang mirip. Kita punya sejumlah angka. Setiap angka ini diambil, akan masuk ke *cache* terlebih dahulu, dan berada di prioritas terbawah. Kita ambil angka berikutnya, jika angka ini sama dengan salah satu angka di dalam *cache*, maka prioritasnya di dalam *cache* akan dinaikkan satu tingkat. Jika sudah berada di prioritas teratas, maka angka ini tetap di prioritas tertinggi. Bila angka tidak ada dalam *cache*, maka dia masuk ke *cache* dengan prioritas terendah. Sedangkan angka dengan prioritas terendah di *cache* akan dihapus dari *cache*. Jika sebuah angka naik prioritasnya, sementara posisi yang akan ditempati sudah ada angka di sana, kedua angka saling berpindah tempat. Pada akhir proses, angka berapakah yang ada di prioritas teratas? Mari kita lihat contoh berikut.

```
------------------
8 4 4 5 3 9 9 6 9 9     <- Angka yang akan diproses
------------------
Angka    P1  P2  P3        Kapasitas cache
------------------        dapat menampung 3 angka
8         8   -   -        P1 adalah cache
4         4   -   -        prioritas terendah
4         -   4   -        P3 adalah cache
5         5   4   -        prioritas tertinggi
3         3   4   -
9         9   4   -
9         4   9   -
6         6   9   -
9         6   -   9
9         6   -   9
------------------
```

Pada akhir proses yang berada di prioritas tertinggi adalah angka 9.

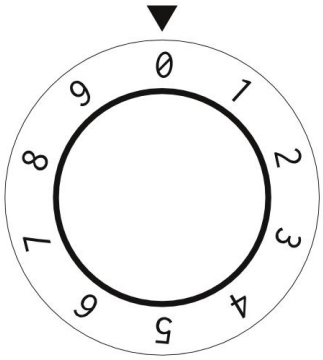
Data soal. Setiap baris berisi sejumlah angka. Angka pertama adalah kapasitas *cache*, kemudian diikuti oleh angka yang akan diproses. Setiap angka dipisahkan oleh satu spasi. Kapasitas minimal cache adalah 3, dan maksimal 100. Angka yang diproses

minimal berjumlah 1 dan maksimal 10^6. Hasil dari program adalah menampilkan angka yang berada di prioritas teratas dalam *cache* setelah semua angka diproses. Jika tidak ada angka di prioritas teratas, maka tampilkan kata NULL.

```
3 7 8 7 9                               h#1: NULL
5 1 1 1 1 1 1 7                         h#2: 1
3 1 2 3 4 5 5 5 8 9                     h#3: 5
```

4.55 Kunci Putar

Sebuah tombol putar dengan diameter seukuran gelas. Angka tertera di sekelilingnya. Membuka kunci ini dengan memutar sampai mencapai kombinasi angka tertentu. Model seperti ini sangat akrab dengan kunci brankas.



Posisi tanda angka ada di atas. Tombol bisa diputar ke kanan maupun ke kiri. Memilih angka dengan cara memutar tombol sampai angka yang dikehendaki berada di posisi atas. Kemudian berhenti sesaat. Lanjutkan ke angka berikutnya. Lanjutannya bisa ke kiri atau ke kanan. Ketentuannya adalah ke arah yang putarannya lebih sedikit. Jika jaraknya sama, maka pilihannya adalah melanjutkan arah yang sama dengan sebelumnya. Putaran pertama selalu berlawanan dengan arah jarum jam (ke kiri).

Kita akan membuat susunan angka: `4350`. Pertama kali kita akan memilih angka 4. Karena ini putaran pertama, maka arahnya ke kiri. Setelah itu putar ke kanan sampai angka 3. Dari angka 3 ke angka 5, putar ke arah kiri. Angka terakhir yaitu 0. Jarak dari angka 5 ke angka 0 sama, baik ke kiri maupun ke kanan. Maka dipilih arah yang sama dengan sebelumnya, yaitu ke kiri. Untuk mendapatkan kombinasi angka ini, maka gerakannya adalah: `kiri kanan kiri kiri`.

Soal kita adalah berapa kali pertukaran arah dilakukan? Dua kali. Pertukaran arah pertama ketika akan memilih angka 3, dari kiri ke kanan. Pertukaran kedua adalah ketika memilih angka 5, dari arah kanan ke kiri. Untuk angka terakhir, yaitu 0, tidak ada pertukaran arah. Sebelumnya ke kiri, tetap dilanjutkan memutar ke arah kiri.

Data soal. Setiap baris berisi kombinasi angka yang akan didapatkan dengan cara memutar tombol ini. Tidak ada pemisah antar angka. Panjang angka minimal adalah 1 angka, dan maksimal 100 angka. Program dibuat untuk mencari berapa kali terjadi pertukaran arah untuk mendapatkan kombinasi angka.

```
12345                                   h#1: 0
2121                                    h#2: 3
981                                     h#3: 2
```

4.56 Kiri Kanan

Vim adalah sebuah perangkat lunak untuk menyunting teks. Editor ini bekerja dalam banyak mode. Salah satu mode diberi nama mode normal. Dalam mode normal, menekan sebuah huruf pada papan ketik mempunyai arti tertentu. Bisa berpindah posisi, menghapus, menyalin, dan lain sebagainya. Kita akan fokus pada pergerakan saja, khsusus hanya untuk satu baris. Sehingga perpindahan kursor hanyalah ke kiri atau ke kanan.

Menekan huruf `h` akan menyebabkan gerakan ke kiri satu karakter. Jika sudah berada di awal baris, maka akan tetap di sana. Huruf `l` artinya bergerak ke kanan satu karakter. Jika sudah sampai di ujung baris, menekan huruf `l` tidak akan menyebabkan perpindahan kursor.

Huruf `w` artinya ke huruf pertama kata berikutnya. Jika sudah berada di kata terakhir dalam baris tersebut, maka kursor tidak bergerak. Menekan huruf `b` artinya pindah ke huruf pertama kata. Jika kursor sedang berada pada sebuah huruf dalam kata, maka pindah ke huruf pertama kata. Jika sedang berada di huruf pertama kata, maka pindah ke huruf pertama kata di sebelah kirinya. Jika sudah berada di awal baris, kursor tidak bergerak.

Menerapkan kombinasi 4 huruf ini kita akan memindahkan kursor dalam satu baris. Setelah melakukan serangkaian gerakan tersebut di posisi manakah kursor berada? Huruf pertama dalam baris diberi nomor 1. Contohnya sebagai berikut.

```
     Bahwa dalam suatu perjuangan kita harus
l : bahwa dalam suatu perjuangan kita harus
l : bahwa dalam suatu perjuangan kita harus
w : bahwa dalam suatu perjuangan kita harus
w : bahwa dalam suatu perjuangan kita harus
l : bahwa dalam suatu perjuangan kita harus
l : bahwa dalam suatu perjuangan kita harus
b : bahwa dalam suatu perjuangan kita harus
b : bahwa dalam suatu perjuangan kita harus
```

Kombinasi gerakan pada contoh di atas adalah `llwwllbb`. Posisi akhir kursor adalah di karakter nomor 7.

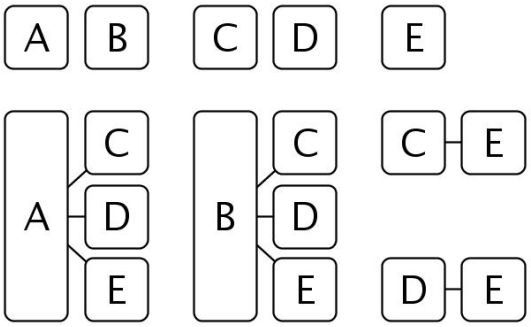
Data soal. Setiap baris berisi kombinasi huruf yang ditekan, diikuti dengan sebuah kalimat yang terdiri dari sejumlah kata. Pemisah antar kata adalah satu buah spasi. Panjang kombinasi gerakan minimal 1 huruf, dan maksimal 100 huruf. Panjang minimal sebuah kalimat adalah 1 kata, dan maksimal 100 kata. Hasil dari program adalah posisi akhir kursor setelah kombinasi gerakan dikerjakan pada baris tersebut.

```
ll aku mau hidup seribu tahun lagi  h#1: 3
b robohnya surau kami               h#2: 1
wh kaki yang terhormat              h#3: 5
```

4.57 Bersalaman

Sebuah acara pertemuan keluarga. Ada banyak orang yang hadir di sana. Ada yang datang bersama pasangannya, dan ada yang datang sendiri. Setiap orang saling bersalaman, kecuali dengan pasangannya sendiri. Berapa kalikah terjadinya tindakan bersalaman dalam acara tersebut?

Contoh yang di sebelah ini dengan jumlah 5 orang A, B, C, D, dan E. Empat orang berpasangan (2 pasangan) yaitu A dengan B dan C dengan D. Dengan melihat diagram, dapat diketahui ada 8 kali orang bersalaman dalam pertemuan tersebut.



Data soal. Setiap baris berisi 2 angka yang dipisahkan oleh satu spasi. Angka pertama adalah jumlah total orang yang ada dalam ruangan. Angka kedua adalah jumlah yang berpasangan. Jumlah orang minimal adalah 2, dan maksimal 1000. Hasil dari program adalah jumlah berapa kali orang bersalaman dalam ruangan itu.

```
4 0                                     h#1: 6
6 2                                     h#2: 14
5 4                                     h#3: 8
```

4.58 Ular Tangga

Permainan ular tangga adalah mainan untuk anak-anak. Dalam perjalanan menuju kotak terakhir ada kotak berisi tangga yang menguntungkan. Apabila berhenti di kotak berisi tangga, pemain bisa meloncat ke kotak yang lebih tinggi. Ada kotak yang berisi ular. Jika berhenti di kotak ini akan menyebabkan pemain turun ke kotak yang lebih rendah.

Papan ular tangga terdiri dari 100 kotak. Diberi nomor dari 1 sampai 100. Sedikit berbeda dengan ular tangga umumnya, jika berhenti di kotak berisi tangga, maka pemain langsung menuju ke kotak terakhir (kotak nomor 100). Jika berhenti pada kotak berisi ular, maka kembali ke kotak nomor 1.

Setiap kotak bernomor kelipatan 7 berisi ular, yaitu kotak dengan nomor 7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98. Sedangkan kotak yang bernomor kelipatan 17 berisi tangga, yaitu kotak dengan nomor 17, 34, 51, 68, 85.

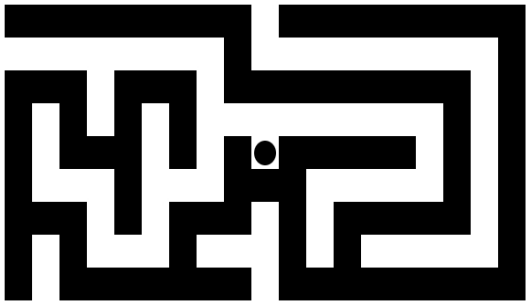
Permainan dimulai dari luar papan. Belum ada yang masuk ke salah satu kotak. Berapa langkah akan dilakukan, tergantung hasil lemparan dadu. Angka yang mungkin muncul dari dadu adalah dari 1 sampai 6. Pemain boleh masuk ke papan dan berjalan jika sudah pernah mendapatkan angka 6. Jika lemparan dadu belum pernah dapat angka 6, maka pemain tetap berada di luar arena. Berikut ini adalah uraian perjalanan apabila mendapatkan angka dadu `3 6 5 3 4`. Di kotak nomor berapakah posisi terakhir pemain?

```
3 : Masih di luar papan permainan
6 : Ke kotak nomor 6
5 : Ke kotak nomor 11
3 : Ke kotak nomor 14. Ada ular, kembali ke kotak nomor 1
4 : Ke kotak nomor 5 (posisi terakhir)
```

Data soal. Setiap baris berisi sejumlah angka. Setiap angka dipisahkan oleh satu spasi. Angka ini adalah hasil lemparan dadu. Sesuai deskripsi tentang permainan ular tangga dalam soal ini, tentukanlah di kotak nomor berapa posisi akhir pemain. Jumlah angka minimal 1 angka, maksimal 100 angka. Hasil lemparan dadu dalam rentang 1 sampai 6. Jika pemain sudah sampai di kotak terakhir (kotak nomor 100), permainan selesai. Tidak perlu dilanjutkan ke angka berikutnya. Apabila pemain masih berada di luar papan, tampilkan angka 0.

```
6 1 1                                   h#1: 2
5 5                                     h#2: 0
6 6 5 4 3 5 5 5                         h#3: 100
```

4.59 Jalan Keluar



Labirin adalah sebuah kerumitan. Jalurnya berbelit-belit. Di atas kertas bisa jadi mainan. Menemukan jalan keluar di dalam labirin bukanlah perkara mudah. Kita akan melakukannya. Dengan posisi awal di tengah-tengah, kita akan mencari adakah jalan keluar dari perangkap labirin ini. Gambar di samping adalah contoh sebuah labirin. Tanda bulat menunjukkan posisi di tengah. Posisi awal selalu dimulai dari sana.

Diagram labirin dapat disusun menggunakan huruf. Dinding diwakili oleh huruf X, sedangkan jalan memakai huruf H. Gambar labirin di atas dikonversi dalam bentuk sebagai berikut.

```
X X X X X X X X H X X X X X X X X X X
H H H H H H H H X H H H H H H H H H X
X X X H X X X H X X X X X X X X X H X
X H X H X H X H H H H H H H H H X H X
X H X X X H X H X O X X X X X X H X H X
X H H H X H H H X X X H H H H H H X H X
X X X H X H X X X H X H X X X X X H X
X H X H H H X H H H X H X H H H H H X
X H X X X X X X X H X X X X X X X X X
```

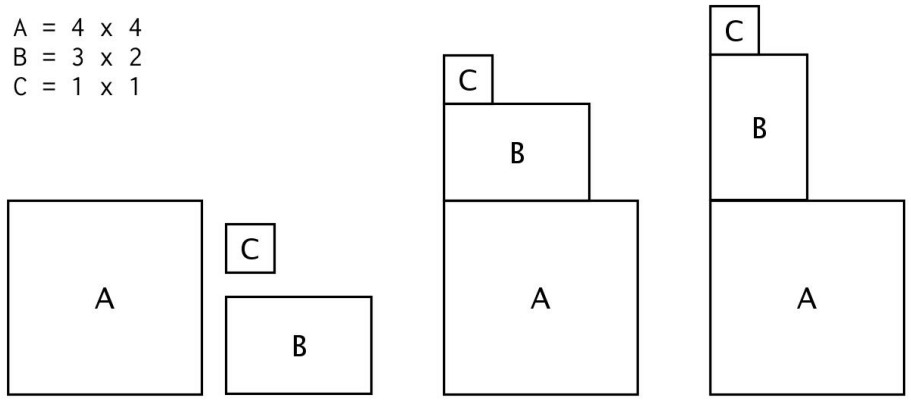
Data soal. Setiap baris berisi data struktur labirin yang terdiri dari huruf X dan H. X menandakan dinding, dan H adalah jalan yang bisa dilalui. Huruf O adalah posisi awal, yaitu di tengah labirin. Matriks untuk labirin ini berupa bujursangkar, artinya panjang dan lebar sama. Sehingga jumlah karakter dalam satu baris merupakan bilangan pangkat 2. Panjang sisi labirin adalah bilangan ganjil.

Setiap satu baris data perlu disusun menjadi matriks n x n. Misalnya ada 25 data, maka disusun menjadi matriks 5 x 5. Posisi awal dalam mencari jalan keluar adalah di tengah-tengah. Hasil dari program adalah kata YA untuk menandakan bahwa labirin ini mempunyai jalan keluar, dan kata TIDAK jika labirin tidak mempunyai jalan keluar.

```
XXXXOXXHX                              h#1: YA
XXXXXHHXXXXHOXXXXXXXXXXHXX            h#2: YA
XXXXOXXXX                              h#3: TIDAK
```

4.60 Menara Kotak

Dalam bidang dua dimensi, ada banyak segiempat yang bertebaran. Ukurannya beragam. Semua segiempat ini akan disusun bertumpuk. Anggaplah bentuknya jadi sebuah menara yang menjulang. Segiempat yang di bawah harus lebih lebar atau sama dengan segiempat yang di atasnya. Berapakah ketinggian maksimal menara yang bisa didapatkan? Berikut ini contohnya.



Tiga kotak ini dapat membentuk dua menara yang sesuai dengan aturan bahwa kotak di bawah lebih lebar atau sama dengan kotak di atasnya. Ambil satuan panjang dalam cm, susunan pertama menghasilkan tinggi 7cm. Susunan ke-2 menghasilkan tinggi 8cm. Tinggi maksimal didapat pada susunan ke-2 yaitu 8cm.

Data soal. Setiap baris berisi data tentang kotak yang akan disusun. Ukuran kotak adalah panjang x lebar. Contohnya 3x6. Setiap data kotak dipisahkan oleh satu spasi. Ukuran kotak paling kecil adalah 1x1 dan ukuran terbesar adalah 100x100. Jumlah kotak minimal 1 dan maksimal 1000 kotak. Hasil dari program adalah angka tinggi menara maksimal yang bisa didapat dengan menyusun kotak ini.

```
1x1 2x2 3x3                    h#1: 6
2x2 6x2 8x2                    h#2: 16
1x1 10x10 10x1                 h#3: 21
```

4.61 Undang-Undang Lagu

Kita dengar sebuah dongengan lagi. Tentang sebuah kerajaan antah berantah, dengan sang raja yang berkuasa. Ada aturan di sana tentang mengarang lagu. Dilarang mencipta lagu diluar batasan yang dititahkan paduka raja. Hanya ada tiga komponen yang tersedia. Jenis not, panjang not, dan maksimal jumlah not dalam satu lagu. Jika tiga komponen ini sudah ditetapkan, maka dalam batas itulah sebuah lagu boleh dibuat. Mari kita hitung berapa banyak lagu yang bisa tercipta dari aturan kocak sang raja.

Dari jumlah not yang ditetapkan, tentu saja bebas memilih. Boleh satu not saja, bisa sebagian, boleh juga semuanya. Tentang panjang lagu, selama tidak melebihi batas maksimal, juga tidak masalah. Andaikan batas maksimal adalah 10 not, mencipta lagu dengan satu not tentu saja tidak melanggar aturan. Pernahkah Anda mendengar lagu satu not? Saya belum.

Misalnya not yang tersedia hanya ada 3, yaitu `do re mi`. Panjang not hanya ada 2 jenis, yaitu satu dan setengah. Maksimal sebuah lagu tak lebih dari 2 not. Dengan aturan seperti ini maka ada 42 lagu yang bisa diciptakan.

Data soal. Setiap baris berisi data tentang 3 elemen ini. Angka pertama adalah jumlah not yang dibolehkan, angka kedua adalah panjang not yang ada, dan angka ketiga adalah maksimal jumlah not dalam sebuah lagu. Setiap angka dipisahkan oleh satu spasi. Hasil dari program adalah jumlah maksimal lagu yang bisa diciptakan dari ketentuan tersebut.

```
3 3 2                                   h#1: 90
1 2 1                                   h#2: 2
2 2 2                                   h#3: 20
```

4.62 Warna Dominan

Salah satu model pengkodean warna menggunakan kombinasi tiga warna, yaitu merah (R), hijau (G), dan biru (B). Setiap elemen warna ini punya rentang dari 0 sampai 255. Dalam bilangan heksadesimal mulai dari 00 sampai FF. Warna hitam mempunyai komponen merah, hijau, biru yang sama, yaitu 0. Artinya R = 00, G = 00, dan B = 00. Penulisannya digabung antara R, G, dan B, yaitu `000000`. Sedangkan warna hijau, ditulis dengan `00FF00`. Komponen merah dan biru bernilai 0, dan komponen hijau bernilai 255 atau FF dalam heksadesimal.

Kombinasi tiga komponen ini menghasilkan banyak warna. Diberikan sebuah daftar yang terdiri dari banyak warna. Kita akan mencari warna dominan dalam daftar tersebut. Apa definisi warna dominan dalam soal ini? Misalnya ada satu warna dengan kode `F399A0`. Komponen warna dengan nilai tertinggi adalah komponen R yaitu F3. Artinya warna dominan adalah R (merah). Dalam daftar warna yang diberikan, kita akan putuskan warna apakah yang dominan, contohnya sebagai berikut.

```
004455  3321FE  EE6633  A07689  330010  550055  333333
- - B   - - B   R - -   R - -   R - -   R - B   - - -
```

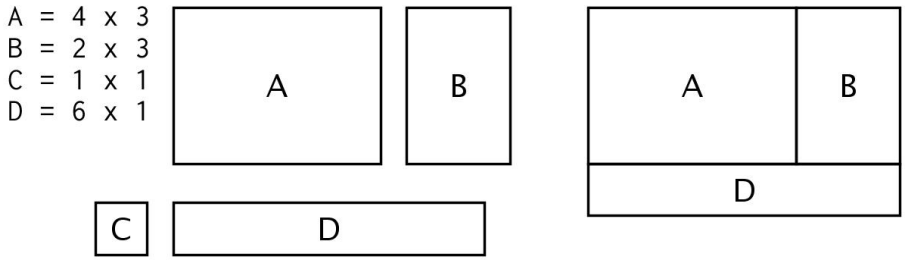
Dalam daftar ada 7 warna. Huruf yang dituliskan di bawah setiap warna menunjukkan komponen yang dominan. Bisa ada lebih dari satu komponen yang dominan, seperti pada warna `550055`. Komponen dominan ada 2 yaitu R dan B. Dalam kasus semua komponen sama, seperti yang terjadi pada warna `333333` maka tidak ada yang dianggap dominan. Dari diagram ini, dominan B ada 3, dan dominan R ada 4. Warna dominan untuk daftar ini adalah R.

Data soal. Setiap baris terdiri dari sejumlah kode warna. Setiap kode warna dipisahkan oleh satu spasi. Jumlah warna minimal 1, maksimal 1000 warna. Kita akan mencari warna apa yang dominan dari semua warna yang diberikan tersebut. Hasil dari program adalah huruf R, G, atau B. Jika tidak ada warna dominan, atau ditemukan lebih dari satu warna dominan, tulis X.

```
333333 444444 555555                    h#1: X
980000 DD0000 FE5689                    h#2: R
000000 00A900 00CC00                    h#3: G
```

4.63 Kotak Kecil Kotak Besar

Kita punya 3 segiempat. Ketiga segiempat ini ukurannya bisa sama, bisa berbeda. Ketiganya akan disusun untuk membentuk segiempat yang lebih besar. Menyusun dengan cara meletakkan secara berdampingan tanpa ada ruang kosong diantara ketiganya. Menyusunnya tidak boleh ada yang berhimpitan. Hasilnya haruslah sebuah segiempat yang lebih besar. Hal ini tentu ada dua kemungkinan. Bisa dibuat segiempat yang lebih besar, atau tidak bisa. Berikut ini contohnya.



Jika 3 kotak yang dimiliki adalah kotak A, B, dan kotak C, kita tidak bisa menyusunnya untuk mendapatkan segiempat yag lebih besar. Kalau 3 segiempat yang dipunyai adalah A, B, dan D, maka bisa dibuat segiempat yang lebih besar.

Data soal. Setiap baris berisi 3 data segiempat. Pemisah setiap data adalah satu spasi. Satu buah segiempat ditulis dengan bentuk panjang x lebar, misalnya 3x2. Hasil dari program adalah kata YA untuk menyatakan bahwa ketiga kotak itu bisa disusun untuk mendapatkan kotak yang lebih besar. Kata TIDAK jika tidak bisa disusun menjadi segiempat yang lebih besar.

```
1x1 1x1 1x1                    h#1: TIDAK
2x3 2x3 4x1                    h#2: YA
3x2 1x1 1x1                    h#3: YA
```

4.64 Yang Tak Berulang

Dalam soal ini kita akan menjumlahkan beberapa angka. Hasil yang diinginkan adalah hasil penjumlahan angka yang masuk dalam kriteria tertentu. Misalnya diberikan angka dalam rentang dari 10 sampai 15. Semua angka yang ada dalam rentang ini dijumlahkan, kecuali angka yang di dalamnya ada angka yang berulang.

```
10    11    12    13    14    15
10          12    13    14    15    => 64

100   101   102   103   104   105
            102   103   104   105   => 414
```

Dari contoh di atas, pada rentang 10 sampai 15, maka angka 11 tidak ikut dijumlahkan, karena angka itu mengandung angka yang berulang yaitu angka 1. Pada contoh kedua, angka 100 tidak ikut dijumlahkan karena ada angka 0 yang berulang. Angka 101 juga tidak, karena angka 1 berulang di dalamnya.

Data soal. Setiap baris berisi dua angka yang dipisahkan oleh satu spasi. Angka pertama adalah angka awal, angka kedua adalah angka akhir. Hasil dari program adalah penjumlahan seperti kondisi yang digambarkan sebelumnya. Angka minimal adalah 1, dan maksimal adalah 10^6.

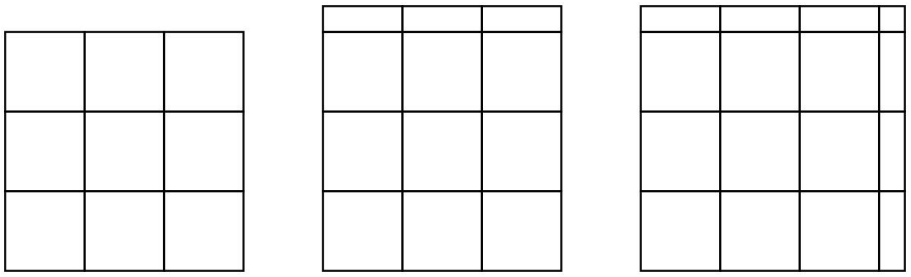
```
10 12                                   h#1: 22
1 3                                     h#2: 6
200 201                                 h#3: 201
```

4.65 Lantai Keramik

Keramik untuk lantai umumnya berbentuk segiempat dengan ukuran tertentu. Misalnya 30cm x 30cm. Ada juga 40cm x 40cm, dan berbagai ukuran lainnya. Sebuah lantai akan dipasang keramik. Kondisinya serba ideal. Tak ada keramik pecah. Jarak antar keramik diabaikan.

Misalnya sebuah lantai berukuran 90cm x 90cm akan dipasang keramik berukuran 30cm x 30xm. Diperlukan 9 buah keramik untuk menutup permukaan lantai ini. Bagaimana dengan lantai berukuran 100cm x 90cm? Ada ruang sebesar 10cm tersisa. Satu buah keramik 30cm x 30cm bisa dipotong menjadi 10cm x 30cm sebanyak 3 buah. Tiga potongan ini dapat menutup ruangan yang tersisa, sehingga total keramik yang diperlukan adalah 10 keramik.

Kita akan menghitung jumlah keramik yang diperlukan untuk melapisi sebuah lantai. Jika ada ruang tersisa yang lebih kecil dari ukuran keramik, maka keramik boleh dipotong. Menutup ruang sisa ini tidak boleh dengan potongan keramik yang lebih kecil dari ruang tersebut. Misalnya ada tersisa sebesar 20cm x 30cm. Tidak boleh mengisinya dengan dua potongan 10cm x 30cm. Gambar berikut ini sebagai ilustrasi pemasangan keramik pada lantai.



90cm x 90cm 90cm x 100cm 100cm x 100cm

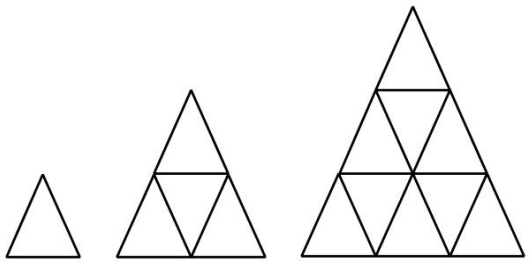
Keramik yang digunakan berukuran 30cm x 30cm. Pada lantai pertama yang berukuran 90cm x 90cm diperlukan 9 buah keramik. Pada lantai kedua yang berukuran 90cm x 100cm diperlukan 10 buah keramik. Satu buah keramik dipotong menjadi tiga buah, masing-masing 10cm x 30cm untuk menutup sisa lantai. Sedangkan pada contoh ketiga diperlukan 12 keramik.

Data soal. Setiap baris berisi dua data, yaitu ukuran keramik dan ukuran lantai. Kedua data ini dipisahkan oleh satu spasi. Ukuran keramik dalam bentuk panjang x lebar, demikian juga dengan ukuran lantai. Ukuran minimal keramik adalah 1x1, dan ukuran maksimal adalah 1000x1000. Ukuran minimal lantai adalah 1x1 dan maksimal 1000x1000. Hasil dari program adalah banyaknya keramik yang diperlukan untuk menutup lantai tersebut.

```
30x30 90x90                                 h#1: 3
1x1 100x100                                 h#2: 10000
2x2 5x5                                     h#3: 7
```

4.66 Batang Korek Api

Teka-teki geometri menggunakan batang korek api kaya dengan imajinasi dan solusi yang kadang tidak sederhana untuk ditemukan. Umumnya memindahkan susunan hingga membentuk pola baru sebagai jawaban.



Dalam soal ini kita juga menggunakan batang korek api untuk membentuk segitiga. Satu buah segitiga memerlukan 3 batang korek api. Sedangkan untuk bentuk kedua, yaitu dengan ketinggian 2 segitiga, diperlukan 9 batang. Bentuk yang ke-3 dengan ketinggian 3 segitiga menghabiskan 18 batang korek api.

Data soal. Setiap baris berisi satu buah angka. Angka ini menunjukkan ketinggian segitiga yang akan dibentuk. Angka minimal adalah 1, dan maksimal 1000. Carilah berapa banyak batang korek api diperlukan untuk membangunnya.

```
1                                       h#1: 3
2                                       h#2: 9
3                                       h#3: 18
```

4.67 Kode Morse Tanpa Spasi

Kode morse pernah digunakan sebagai sarana untuk mengirimkan pesan melalui gelombang radio, melalui cahaya, atau media lain yang mungkin. Kode ini terdiri dari suara panjang dan pendek. Bisa dutulis dengan *biiiiip* untuk suara panjang, dan *bip* untuk suara pendek. Tapi tampaknya kurang praktis. Kita gunakan tanda titik (.) untuk bunyi pendek dan tanda strip (-) untuk bunyi panjang. Sebagai contoh huruf A dapat ditulis dengan `.-`

Berikut ini tabel kode morse untuk huruf A sampai Z.

A	.-	F	..-.	K	-.-	P	.--.	U	..-	Z	--..
B	-...	G	--.	L	.-..	Q	--.-	V	...-		
C	-.-.	H	....	M	--	R	.-.	W	.--		
D	-..	I	..	N	-.	S	...	X	-..-		
E	.	J	.---	O	---	T	-	Y	-.--		

Konversi kata menjadi kode morse dilakukan sesuai tabel di atas. Setiap huruf dipisahkan oleh satu spasi. Kata `IBU` dalam kode morse menjadi `.. -... ..-`

Apabila diberikan kode morse `.. -... ..-` kita dapat mengembalikannya ke bentuk huruf. Tentu saja merujuk ke tabel di atas. Hasilnya adalah `IBU`. Persoalan kita terjadi jika spasi yang memisahkan huruf dalam kode morse itu dihilangkan. Lihat contoh berikut.

Kode morse awal	.. -... ..-	IBU
Kode morse tanpa spasi	..-.....-	?
Kemungkinan jika diberi spasi	.. -... ..-	IBU
	. . -... ..-	EEBU
	. . -... . .-	EEBEA
	.. -.. . ..-	IDEU

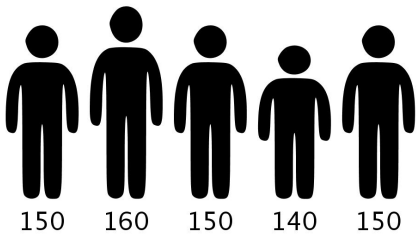
Ada banyak kemungkinan cara meletakkan spasi dalam kode itu. Mengisi spasi tentu harus menghasilkan kode morse yang benar, sesuai yang ada di tabel sebelumnya. Dalam soal ini kita akan mencari ada berapa banyak kemungkinan pengisian spasi dalam sebuah kode morse tanpa spasi.

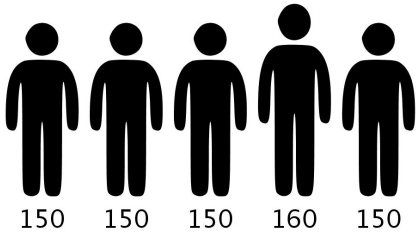
Data soal. Setiap baris berisi kode morse tanpa spasi, yang terdiri dari titik dan tanda strip. Minimal 1 karakter, dan maksimal 20 karakter. Program mencari berapa kemungkinan yang bisa jika dilakukan penyisipan spasi dalam kode tersebut. Apabila kode terdiri dari 1 karakter, maka hasilnya adalah 0.

```
.                                       h#1: 0
--                                      h#2: 1
.-.                                     h#3: 3
```

4.68 Berdiri Tidak Sama Tinggi

Senin pagi dan upacara bendera dan barisan anak sekolah. Pernah sekolah? Pernah upacara? Pernah berbaris dalam upacara? Benarkah murid yang pendek selalu di depan, dan yang tinggi di belakang? Soal ini perihal tinggi badan dalam barisan. Pertanyaan lainnya, biarkan saja. Nggak penting.





Sebuah sekolah punya aturan istimewa. Agar muridnya lebih kreatif menemukan solusi. Lazimnya yang pendek di depan, lalu berurut ke belakang semakin tinggi. Itu biasa. Bapak kepala sekolah punya ide berbeda. Syaratnya sederhana. Siswa dengan tinggi sama dilarang berdekatan. Mari lihat gambar berikut.

Pada barisan pertama bisa disusun agar tak ada siswa dengan tinggi sama berdekatan. Sedangkan barisan kedua tidak mungkin disusun seperti aturan itu.

Data soal. Setiap baris berisi sejumlah angka yang menunjukkan tinggi siswa. Setiap angka dipisahkan satu spasi. Jumlah angka minimal 2, dan maksimal 1000. Hasil dari program adalah kata YA jika barisan dapat disusun tanpa ada tinggi yang sama berdekatan. Kata TIDAK jika tidak bisa disusun sesuai aturan ini.

```
100 120 100 120 120                    h#1: YA
10 10 10 10 10 10 10 10                h#2: TIDAK
1 2 3 4                                h#3: YA
```

4.69 Set dan Shift

Set dan *Shift* adalah dua buah instruksi primitif. Berbekal perintah sederhana ini kita akan menyusun bilangan biner. *Set* kita singkat menjadi `SE`. *Shift* disingkat menjadi `SF`.

`SE` adalah instruksi yang menjadikan bit paling kiri bernilai 1. Jika bit paling kiri sudah bernilai 1, menjalankan instruksi `SE` tidak menimbulkan perubahan. Bit paling kiri tetap bernilai 1. Sedangkan instruksi `SF` menggeser setiap bit satu langkah ke kanan. Bit paling kanan dibuang. Bit baru yang masuk di posisi paling kiri bernilai 0. Berikut ini contoh penggunaan kedua instruksi tersebut pada angka biner 4 bit.

```
---------------------
      MSB       LSB
---------------------
Awal  0   0   0   0
SE    1   0   0   0
SF    0   1   0   0
SF    0   0   1   0
SE    1   0   1   0
SE    1   0   1   0
SF    0   1   0   1
SF    0   0   1   0
---------------------
```

Bit paling kiri adalah MSB (*Most Significant Bit*), dan paling kanan adalah LSB (*Least Significant Bit*). Setelah menjalankan rangkaian instruksi `SE SF SF SE SE SF SF`, didapatkan bilangan biner `0010`, dalam desimal bernilai 2.

Dalam menyusun instruksi bisa terjadi satu instruksi diulang beberapa kali secara berurutan. Dalam contoh di atas, ada `SE` yang berurutan 2 kali, juga `SF` ada yang berurutan 2 kali. Instruksi sama yang berurutan dapat ditulis dengan cara `nXX`. Dimana n adalah sebuah angka, dan XX adalah instruksi. Misalnya `3SE` sama dengan `SE SE SE`. `1SF` sama dengan `SF`.

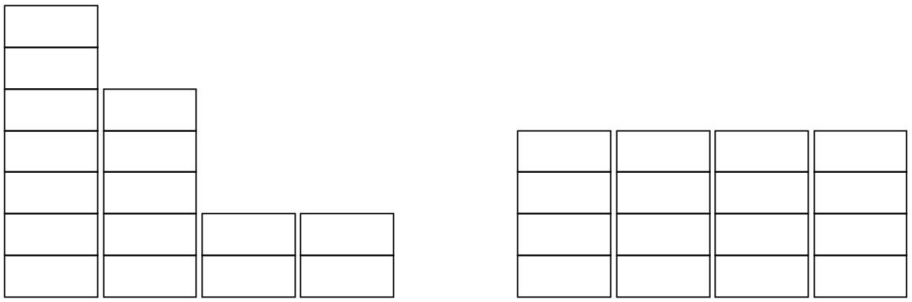
Data soal. Setiap baris berisi data tentang jumlah bit dan sejumlah instruksi. Angka pertama adalah jumlah bit bilangan biner yang akan disusun. Dipisahkan dengan satu spasi, setelah itu adalah rangkaian instruksi untuk menyusun bilangan biner. Setiap instruksi dipisahkan oleh satu spasi. Panjang sebuah bilangan biner minimal adalah 2 bit dan maksimal 24 bit. Jumlah instruksi minimal 1, maksimal 1000 instruksi. Hasil dari program adalah angka (dalam desimal) yang dihasilkan setelah instruksi dijalankan.

```
4 SE SE SF                                  h#1: 4
8 4SF                                       h#2: 0
2 SE SF                                     h#3: 1
```

4.70 Bata Sama Tinggi

Tukang batu menurunkan bata dari bak terbuka. Memindahkan semua ke samping rumah. Kemudian berlalu setelah menerima rupiah. Baginya urusan telah selesai. Sayang, bagi tuan rumah masih menyisakan persoalan. Bata itu disusun tak sama tinggi. Bata yang tersusun rapi dan sama rata adalah obsesi. Jangan diingkari. Demikian maunya pemilik rumah.

Bata akhirnya disusun kembali. Bata yang ada dalam onggokan lebih tinggi dipindahkan satu persatu ke onggokan yang rendah. Tujuannya tentu saja membuatnya sama rata. Banyaknya tumpukan sebelum dan sesudah disusun adalah sama. Agar tidak menguras banyak tenaga, bata yang dipindahkan haruslah sesedikit mungkin. Dalam gambar berikut ini, kelompok sebelah kiri menjadi seperti kelompok sebelah kanan. Jumlah bata yang dipindahkan adalah 4 buah.

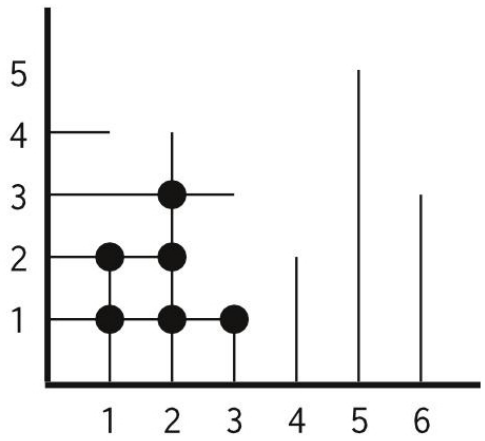


Data soal. Setiap baris berisi sejumlah angka. Setiap angka dipisahkan oleh satu spasi. Angka menunjukkan tinggi tumpukan bata. Angka minimal 1, dan maksimal 1000. Jumlah angka yang menunjukkan banyaknya tumpukan, minimal 2, dan maksimal 1000. Hasil dari program adalah jumlah bata yang dipindahkan untuk mendapatkan tumpukan yang sama tinggi.

```
4 4 4 4 4 4                          h#1: 0
10 12 10 12 10 12 10 12              h#2: 4
20 10                                h#3: 5
```

4.71 Garis yang Bersilangan

Dalam ruang dua dimensi kita menggambar beberapa garis. Garis vertikal dan garis horizontal. Garis-garis itu berbeda panjangnya. Garis berada dalam sebuah koordinat. Ada sumbu x dan sumbu y. Garis berada pada daerah positif. Garis vertikal selalu dimulai dari sumbu x. Garis horizontal selalu dimulai dari sumbu y. Jarak antar titik dalam koordinat adalah 1 cm. Panjang garis dalam satuan cm. Berapa titik pertemuan dan persilangan yang terjadi antara garis vertikal dan horizontal tersebut? Mari kita lihat dalam gambar berikut.



Dari gambar dapat dilihat ada 6 garis vertikal dan 4 garis horizontal. Titik pertemuan/persilangan dari garis-garis ini adalah 6. Data untuk gambar ini dalam bentuk seperti berikut.

```
2 4 1 2 5 3-3 2 3 1
```

Data soal Setiap baris berisi angka yang menunjukkan tinggi/panjang garis. Pemisah antara angka adalah satu buah spasi. Pemisah data antara garis vertikal dan data garis horizontal adalah tanda strip (-). Jumlah garis vertikal minimal 1, dan maksimal 1000. Jumlah garis horizontal minimal 1, dan maksimal 1000. Panjang minimal garis adalah 1cm dan maksimal 1000cm. Hasil dari program adalah jumlah titik pertemuan/persilangan dari garis vertikal dan horizontal.

```
1 1-1 1                                    h#1: 1
1 1-100                                    h#2: 2
2 2 2-10 10 10                             h#3: 6
```

Kodenesia
71 Soal Algoritma Dasar
Yanmarshus Bachtiar

Edisi revisi November 2017
Diterbitkan pertama kali Januari 2014
Diterbitkan secara pribadi oleh Yanmarshus Bachtiar

Perangkat lunak untuk membuat buku ini
Penyunting naskah Vim
Format dokumen LaTeX
Paket pengolah dokumen Texlive
Penangkap tampilan layar ImageMagick
Pengolah gambar digital Gimp
Penyunting diagram yEd
Konversi bitmap ke vector potrace
Penata letak untuk sampul Scribus
Distribusi sistem operasi Linux Slackware