Last Supper

Leonardo sangat aktif saat mengerjakan "Last Supper", lukisan dindingnya yang sangat terkenal: salah satu tugas hariannya adalah menentukan warna-warna yang akan digunakan sepanjang hari itu. Dia membutuhkan banyak warna padahal hanya dapat menaruh sebagian dari keseluruhan warna secara terbatas pada scaffold (tempat menaruh cat). Asistennya bertugas antara lain, naik scaffold untuk membawakan warna yang diminta baginya, dan kemudian turun meletakkannya ke rak (shelf) di lantai.

Pada soal ini, Anda diminta untuk menulis dua program terpisah untuk membantu sang asisten. Program pertama akan menerima instruksi-instruksi dari Leonardo (sebuah sekuens dari warna-warna yang dibutuhkan Leonardo pada siang hari), dan membuat sebuah string bit yang *pendek*, yang disebut *advice*. Selama memproses permintaan-permintaan Leonardo pada siang hari, sang asisten tidak akan dapat mengakses permintaan-permintaan Leonardo selanjutnya, hanya ke advice yang dihasilkan oleh program pertama Anda. Program kedua akan menerima advice, dan kemudian menerima dan memproses permintaan-permintaan Leonardo secara online (yakni, satu pada satu saat). Program ini harus mampu untuk memahami apa arti dari advice dan memanfaatkannya untuk memilih secara optimal. Semuanya dijelaskan dengan lebih detail di bawah ini.

Memindahkan warna antara rak dan scaffold

Kita akan mempertimbangkan sebuah skenario yang disederhanakan. Andaikan ada N warna yang dinomori dari 0 hingga N - 1, dan setiap hari Leonardo meminta sang asisten sebuah warna baru sebanyak tepat N kali. Anggap C adalah sekuens dari N warna yang diminta oleh Leonardo. Maka, kita dapat berpikir bahwa C sebagai sekuens dari N bilangan, setiap bilangan bernilai antara 0 dan N - 1, inklusif. Perhatikan bahwa beberapa warna mungkin tidak ada sama sekali dalam C, dan warna lain mungkin muncul beberapa kali.

Scaffold selalu penuh dan berisi beberapa K dari N warna, dengan $K < N$. Pada awalnya, scaffold berisi warna dari 0 hingga K - 1, inklusif.

Sang asisten memroses permintaan Leonardo satu permintaan pada satu saat. Apabila warna yang diminta *sudah ada di scaffold*, sang asisten dapat beristirahat. Bila tidak, dia harus mengambil warna yang diminta dari rak dan memindahkannya ke scaffold. Tentu saja, tak ada ruang dalam scaffold untuk warna yang baru diminta sehingga sang asisten harus memilih satu warna pada scaffold dan mengembalikan dari scaffold ke rak.

Strategi optimal Leonardo

Sang asisten menginginkan beristirahat sebanyak kali yang mungkin. Banyaknya permintaan yang menyebabkan asisten dapat beristirahat tergantung kepada pilihannya selama proses terjadi. Secara

lebih tepat, setiap kali sang asisten harus mengambil sebuah warna dari scaffold, pilihan lain dapat memberikan hasil yang berbeda selanjutnya. Leonardo menjelaskan kepadanya bagaimana cara mencapai tujuan tersebut dengan mengetahui C. Pilihan warna terbaik yang diambil dari scaffold diperoleh dengan mempelajari warna-warna yang sedang ada pada scaffold, dan warna yang diminta selanjutnya di C. Sebuah warna sebaiknya dipilih dari scaffold sesuai aturan sebagai berikut:

- Jika ada sebuah warna pada scaffold yang tidak akan pernah diperlukan lagi selanjutnya, maka sang asisten harus mengambil warna itu dari scaffold.
- Bila tidak, warna yang dihapus dari scaffold sebaiknya merupakan *yang berikutnya paling lama (furthest) akan dibutuhkan*. (Yaitu, untuk setiap warna pada scaffold kita akan menemukan *first future occurence*-nya. Warna yang dikembalikan ke rak adalah yang akan digunakan terakhir.)

Dapat dibuktikan bahwa apabila strategi Leonardo digunakan, sang asisten akan beristirahat sebanyak kali mungkin.

Contoh 1

Anggap N = 4, dengan demikian kita mempunyai 4 warna (dinomori dari 0 hingga 3) dan ada 4 permintaan. Anggap sekuens permintaan adalah C = (2, 0, 3, 0). Juga, asumsikan bahwa K = 2. Yakni, Leonardo mempunyai sebuah scaffold yang mampu menampung 2 warna setiap saat. Seperti yang disebutkan di atas, scaffold pada awalnya berisi hanya warna 0 dan 1. Kita akan menuliskan isi scaffold sebagai berikut: [0, 1]. Salah satu kemungkinan bahwa sang asisten dapat menangani permintaan tersebut adalah sebagai berikut:

- Warna pertama yang diminta (nomor 2) tidak ada pada scaffold. Sang asisten meletakkan di sana dan memutuskan untuk menghilangkan warna 1 dari scaffold. Isi scaffold adalah [0, 2].

- Permintaan berikutnya adalah warna (nomor 0) yang sudah ada pada scaffold, maka sang asisten dapat beristirahat.

- Untuk permintaan ketiga (nomor 3), sang asisten mengeluarkan warna 0, mengubah scaffold menjadi [3, 2].

- Akhirnya, warna yang diminta terakhir kali (nomor 0) harus diambil dari rak ke scaffold. Sang asisten memutuskan untuk menghilangkan warna 2, dan scaffold sekarang menjadi [3, 0].

Perhatikan bahwa pada contoh di atas, sang asisten tidak mengikuti strategi optimal Leonardo. Strategi optimal seharusnya adalah menghapus warna 2 pada langkah ketiga sehingga asisten dapat beristirahat lagi pada langkah terakhir.

Strategi sang asisten jika ingatannya terbatas

Pada pagi hari, sang asisten meminta Leonardo untuk menulis C pada secarik kertas sehingga ia dapat memahami dan mengikuti strategi optimal. Namun, Leonardo terobsesi untuk merahasiakan teknik pekerjaannya, sehingga ia menolak asistennya mendapatkan kertas tersebut. Dia hanya memperbolehkan asistennya membaca C dan memintanya untuk mengingatnya.

Sayangnya, ingatan sang asisten sangat buruk. Dia hanya bisa mengingat hingga M bit. Secara umum, ini bisa mengakibatkan dia tidak bisa merekonstruksi ulang seluruh sekuens C. Sehingga, sang asisten harus menemukan sebuah cara cerdas untuk menentukan sekuens bit yang akan ia ingat. Kita akan menyebut sekuens ini sebagai *sekuens advice* dan kita akan menyatakan ini sebagai A.

Contoh 2

Pada pagi hari, sang asisten bisa mengambil kertas Leonardo yang berisi sekuens C, membaca sekuens itu, dan membuat semua pilihan yang diperlukan. Salah satu cara yang bisa ia lakukan adalah mempelajari state dari scaffold setelah setiap permintaan. Sebagai contoh, ketika menggunakan strategi (sub-optimal) yang diberikan pada Contoh 1, sekuens dari state scaffold itu adalah [0, 2], [0, 2], [3, 2], [3, 0]. (Ingat bahwa dia mengetahui bahwa state inisial dari scaffold itu adalah [0, 1].)

Sekarang asumsikan bahwa kita memiliki M = 16, sehingga sang asisten dapat mengingat informasi hingga 16 bit. Untuk N = 4, kita bisa menyimpan setiap warna menggunakan 2 bit. Oleh karena itu, 16 bit cukup untuk menyimpan sekuens dari state scaffold di atas. Jadi, sang asisten menentukan sekuens advice berikut ini: A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`).

Pada siang harinya, sang asisten dapat men-decode sekuens advice ini dan menggunakannya untuk menentukan pilihannya.

(Tentu saja, dengan M = 16 sang asisten juga bisa mengingat seluruh sekuens C dengan hanya menggunakan 8 dari 16 bit yang tersedia. Pada contoh ini kami hanya ingin mengilustrasikan bahwa dia bisa memiliki pilihan lain, tanpa memberikan solusi yang baik.)

Statement

Anda diharapkan untuk menulis *dua program terpisah* dengan menggunakan bahasa pemrograman yang sama. Program-program ini akan dieksekusi secara sekuensial, tanpa dapat berkomunikasi satu sama lain selama eksekusi.

Program yang pertama harus merupakan program yang digunakan sang asisten di pagi hari. Program ini akan memperoleh sekuens C, dan harus menentukan sekuens advice A.

Program kedua harus merupakan program yang digunakan sang asisten pada siang hari. Program ini harus menerima sekuens advice A, kemudian memproses sekuens C yang merupakan permintaan-permintaan Leonardo. Perhatikan bahwa sekuens C hanya akan diberikan pada program ini satu permintaan setiap saat, dan setiap permintaan harus diproses sebelum menerima permintaan berikutnya.

Lebih tepatnya, pada program pertama Anda harus mengimplementasi sebuah routin `ComputeAdvice`(C, N, K, M) yang memiliki sebagai input adalah array C yang terdiri dari N bilangan bulat (masing-masing berada pada 0, …, N - 1), sebuah bilangan K yang menyatakan banyak warna pada scaffold, dan bilangan M yang menyatakan banyaknya bit yang tersedia untuk advice. Program ini harus menentukan sebuah sekuens advice A yang terdiri dari hingga M bit. Program tersebut kemudian harus memberikan sekuens A ke sistem, untuk setiap bit dari A sesuai

urutannya, dengan memanggil rutin berikut ini.

- `WriteAdvice`(B) — tambahkan bit B ke sekuens advice A saat ini. (Anda dapat memanggil rutin ini paling banyak M kali.)

Pada program kedua Anda harus mengimplementasi sebuah routine `Assist`(A, N, K, R). Masukan dari routine ini adalah sekuens advice A, bilangan bulat N dan K seperti yang dijelaskan di atas, dan panjang yang sebenarnya, R, dari sekuens A dalam bit ($R \leq M$). Routine ini harus mengeksekusi strategi yang Anda usulkan untuk sang asisten, menggunakan routine-routine berikut ini yang telah disediakan untuk Anda:

- `GetRequest`() — mengembalikan warna berikutnya yang diminta oleh Leonardo. (Informasi mengenai permintaan-permintaan setelahnya belum diberikan.)

- `PutBack`(T) — kembalikan warna T dari scaffold ke rak. Anda hanya boleh memanggil routine ini dengan T merupakan salah satu warna yang sedang berada pada scaffold.

Ketika dieksekusi, routine `Assist` yang Anda buat harus memanggil `GetRequest` sebanyak tepat N kali, yang masing-masing menerima sebuah permintaan Leonardo, secara berurutan. Setelah setiap pemanggilan `GetRequest`, jika warna yang ia kembalikan *tidak* berada pada scaffold, Anda *harus* juga memanggil `PutBack`(T) dengan T yang Anda pilih. Selain itu, Anda *tidak boleh* memanggil `PutBack`. Kegagalan dalam melakukan ini dianggap sebagai error dan akan menyebabkan program Anda berakhir. Harap diingat bahwa pada awalnya scaffold berisi warna-warna dari 0 hingga K - 1, inklusif.

Suatu test case khusus akan dianggap diselesaikan jika dua buah routine yang Anda buat sesuai dengan semua konstrain yang diberikan, dan banyaknya call ke `PutBack` adalah *sama persis* dengan hasil dari strategi optimal Leonardo . Perhatikan bahwa jika ada banyak strategi yang dapat mencapai banyak pemanggilan yang sama ke `PutBack`, program Anda dibolehkan untuk menjalankan yang manapun (yaitu, tak harus mengikuti strategi optimal Leonardo, jika ada strategi lain yang sama baiknya.)

Contoh 3

Melanjutkan Contoh 2, asumsikan bahwa dalam `ComputeAdvice` Anda menentukan A = (`0`, `0`, `1`, `0`, `0`, `0`, `1`, `0`, `1`, `1`, `1`, `0`, `1`, `1`, `0`, `0`).Untuk mengkomunikasikannya ke sistem, Anda diharapkan melakukan deretan pemanggilan sebagai berikut: `WriteAdvice(0)`, `WriteAdvice(0)` , `WriteAdvice(1)` , `WriteAdvice(0)` , `WriteAdvice(0)`, `WriteAdvice(0)` , `WriteAdvice(1)` , `WriteAdvice(0)` , `WriteAdvice(1)`, `WriteAdvice(1)` , `WriteAdvice(1)` , `WriteAdvice(0)` , `WriteAdvice(1)`, `WriteAdvice(1)`, `WriteAdvice(0)`, `WriteAdvice(0)`.

Routine Anda yang kedua `Assist` kemudian akan dieksekusi, menerima sekuens A di atas, dan nilai N = 4, K = 2, dan R = 16. Routine `Assist` kemudian harus melakukan tepat N = 4 calls ke `GetRequest`. Juga, setelah beberapa dari permintaan-permintaan tersebut, `Assist` seharusnya akan memanggil `PutBack`(T) dengan pilihan T yang sesuai.

Tabel di bawah menunjukkan deretan pemanggilan yang sesuai dengan pilihan-pilihan (sub-optimal) pada Contoh 1. Tanda "hyphen" ('-') menunjukkan tak ada pemanggilan ke `PutBack`.

`GetRequest()`	Action
2	`PutBack`(1)
0	-
3	`PutBack`(0)
0	`PutBack`(2)

Subtask 1 [8 points]

- $N \leq 5\,000$.
- Anda dapat menggunakan paling banyak M = 65 000 bits.

Subtask 2 [9 points]

- $N \leq 100\,000$.
- Anda dapat menggunakan paling banyak M = 2 000 000 bits.

Subtask 3 [9 points]

- $N \leq 100\,000$.
- $K \leq 25\,000$.
- Anda dapat menggunakan paling banyak M = 1 500 000 bits.

Subtask 4 [35 points]

- $N \leq 5\,000$.
- Anda dapat menggunakan paling banyak M = 10 000 bits.

Subtask 5 [up to 39 points]

- $N \leq 100\,000$.
- $K \leq 25\,000$.
- Anda dapat memakai paling banyak M = 1 800 000 bits.

Nilai dari subtask ini tergantung kepada panjang advice, R, dari program yang Anda komunikasikan. Lebih tepatnya, jika R_{max} adalah maksimum (dari seluruh test case) dari panjang sekuens advice yang dihasilkan oleh routine `ComputeAdvice`, nilai Anda adalah:

- 39 points jika $R_{max} \leq 200\,000$;

- 39 (1 800 000 - R_{max}) / 1 600 000 points jika 200 000 < R_{max} < 1 800 000;
- 0 points jika $R_{max} \geq 1\ 800\ 000$.

Detail Implementasi

Anda harus mengumpulkan kedua file *dalam bahasa pemrograman yang sama*.

File pertama dinamakan `advisor.c`, `advisor.cpp` atau `advisor.pas`. File ini harus mengimplementasikan routine `ComputeAdvice` seperti dijelaskan di atas dan memanggil routine `WriteAdvice`. File kedua dinamakan `assistant.c`, `assistant.cpp` atau `assistant.pas`. File ini harus mengimplementsikan routine `Assist` seperti dijelaskan di atas dan dapat memanggil routine `GetRequest` dan `PutBack`.

Signature dari semua routines adalah sebagai berikut.

Program C/C++

```
void ComputeAdvice(int *C, int N, int K, int M);
void WriteAdvice(unsigned char a);
```

```
void Assist(unsigned char *A, int N, int K, int R);
void PutBack(int T);
int GetRequest();
```

Program Pascal

```
procedure ComputeAdvice(var C : array of LongInt; N, K, M : LongInt);
procedure WriteAdvice(a : Byte);
```

```
procedure Assist(var A : array of Byte; N, K, R : LongInt);
procedure PutBack(T : LongInt);
function GetRequest : LongInt;
```

Routines harus berperilaku seperti yang dijelaskan di atas. Tentu saja, Anda boleh mengimplementasikan routine lain untuk keperluan internal mereka. Untuk program C/C++, routines internal Anda harus dideklarasikan `static`, karena sample grader akan me-link semuanya bersama-sama. Alternatif lainnya, hindari mempunyai dua routine (satu di setiap program) dengan nama yang sama. Program yang Anda kumpulkan tidak boleh berinteraksi dengan cara apapun melalui standard input/output, maupun melalui file lain.

Saat memrogram solusi Anda, Anda harus memperhatikan instruksi sebagai berikut (templates pada contest environment memenuhi syarat yang didaftarkan sebagai berikut)

Program C/C++

Pada awal dari program solusi Anda, Anda harus meng-include file `advisor.h` pada advisor dan `assistant.h` pada assistant. Caranya adalah dengan memasukkan baris sebagai berikut pada source code Anda:

```
#include "advisor.h"
```

atau

```
#include "assistant.h"
```

Dua buah file `advisor.h` dan `assistant.h` akan disediakan untuk Anda pada sebuah direktori pada contest environment Anda, dan juga disediakan pada Web interface kontes. Bagi Anda juga akan disediakan (lewat *channel' yang sama) kode dan script untuk mengkompilasi dan menguji solusi Anda. Khususnya, setelah menyalin solusi Anda ke direktori dengan menggunakan script tersebut, Anda harus menjalankan* `compile_c.sh` *atau* `compile_cpp.sh` *(tergantung kepada bahasa yang Anda gunakan untuk menulis kode).*

Program Pascal

Anda harus menggunakan unit `advisorlib` untuk advisor dan `assistantlib` untuk asisten. Hal ini dilakukan dengan memasukkan baris berikut pada source code Anda:

```
uses advisorlib;
```

atau

```
uses assistantlib;
```

Kedua file `advisorlib.pas` dan `assistantlib.pas` akan tersedia bagi Anda dalam sebuah directori di contest environment Anda dan juga akan tersedia pada Web interface kontes. Bagi Anda juga akan disediakan (melalui "channel" yang sama) dengan code dan scripts untuk mengkompilasi dan menguji solusi Anda. Khususnya, setelah menyalin solusi Anda ke directori dengan scripts ini, Anda harus menjalankan `compile_pas.sh`.

Contoh grader

Contoh grader akan menerima input dengan format sebagai berikut:

- baris 1: N, K, M;
- baris 2, …, N + 1: C[i].

Pertama-tama, grader akan mengeksekusi perintah routine `ComputeAdvice`. Ini akan menghasilkan sebuah file `advice.txt`, berisi bit-bit dari sekuens yang sedang aktif, dipisahkan oleh spasi dan diakhiri dengan sebuah 2.

Kemudin ia akan mengeksekusi `Assist` routine Anda, dan membangkitkan output di mana setiap barisnya adalah antara "`R [number]`", atau dalam bentuk "`P [number]`". Baris-baris tipe pertama menunjukkn calls ke `GetRequest()` dan balasan yang diterima. Baris-baris tipe kedua merepresentasikan pemanggilan ke `PutBack`() dan warna yang dipilih untuk dikembalikan. Output diakhiri dengan sebuah baris berbentuk "`E`".

Perhatikan bahwa pada grader ofisial waktu eksekusi dari program Anda mungkin sedikit berbeda dari komputer lokal. Perbedaan ini seharusnya tidak akan berarti. Namun, Anda diundang untuk menggunakan test interface untuk memeriksa apakah solusi Anda bekerja dalam batasan waktu.

Batas Waktu dan Memori

- Batas waktu: 7 detik.
- Batas memori: 256 MiB.