Last Supper

خلال عمل ليونرادو على لوحة العشاء الاخير كان يقف على سقالة وكان يضع مجموعة من الالوا على هذه السقالة زكان مساعده يعطيه بعض هذه الالوان وياخد البعض الاخر ويضعها على الرف في المستودع وخلال هذه العملية كان يضطر الى الصعود والنزول على درجات السلوم الموصول بالسقالة عدة مرات خلال اليوم

في هذه المهمة يبوجب عليك كتابة برنامجين منفصلين لمساعدة مساعد ليوناردو. البرنامج الاول يستلم تعليمات اليوناردو (تسلسل الالوان التي يطلبها اليوناردو خلال اليوم)ولايجاد سلسلة من الحروف القصيرة المسمى نصيحة (advice) وخلال تنفيد طلبات اليوناردو خلال اليوم لن يكون للمساعد القدرة على معرفة طلبات اليوناردو الجديدة الا من خلال النصيحة التي يقدمها برنامجك الاول. البرنامج الثاني سيستلم النصيحة وسينفد طلبات اليوناردو على نحو فوري مثال كل طلب على حده هذا يجب البرنامج يجب ان يكون له القدرة على فهم معنى النصيحة ويستخدمها في تحديدالخيار الامثل وكل شي موضح بالتفاصيل ادناه

نقل الالوان من بين السقالة ورفوف التخزين

دعنا نفترض التصور البسيط الثالي افترض انه هناك N لون مرقمة من 0 الى N-1 وانه كل يوم يطلب ليوناردو من مساعده لون جديد بعدد N مره, دع C تمثل عدد مرات الالوان التي يطلبها ليوناردو من اللون N وبناء عليه دعنا نفترض انه C عدد مرات طلب N وكل منها محصور مابين 0 الى (N-1)ولاحظ انه بعض الالوان يمكن ان لا تقع بين مجموعة C على الاطلاق والبعض الاخر يمكن ان يظهر عدة مرات

السقالة دائما مملوئة وتحتوي K لون من عدد الالوان N (يعني دائما K < N) وبشكل مبدئي تحتوي على الالوان من 0 الى K - 1

المساعد ينفد طلب واحد ليوناردو في كل مره وعندما يكون اللون المطلوب موجود على السقالة حينها يستطيع المساعد الاستراحة والا يتوجب عليه ات يختار اللون المطلوب من المستودع وينقله الى اليقالة وبالطبع لن يكون هناك مجال لترتيب هذا اللون الجديد على السقالةولذا يتوجب على المساعد ان يختار احد الالوان الموجودة على السقالة ويردها الى المستودع

استراجية ليوناردو المثلى

المساعد يريد ان يستريح اكثر مايمكن من المرات عدد الطلبات التي يستطيع ان يرتاح منها تعتمد على اختياره خلال عملية المناولة وبالتحديد كل مرة يقوم المساعد بازالة احد الالوان على السقالة فالخيار المختلف ان يقود الى نتائج مختلفة في المستقبل وليوناردو يشرح له كيف يمكن ان يحقق اهدافه بعرفة اللون C الخيار الافضل للون الذي سيتم ازالته على السقالة يتم من خلال فحص الالوان الموجودة حاليا على السقالة وبقية الالوان المطلوبة من C واللون الذي سيتم اختباره نت السقالة يكون حسب الشروط الثالية : اذا كان هناك لون على السقالة لا يمكن طلبه في المستقبل فان المساعد سوف ينزل هذا اللون من السقالة. والا سيكون هذا اللون الذي تمت ازالته هو اللون المطلوب ثانيةوهكذا في لون يتواجد على السقالة يجب ان نعرف متى سيطلب في الميتقبل. واللون الذي تمارجاعه الى الرف في المخزن هو الذي سنحتاجه في المستقبل. وهكذا سيتم اثبات انه عند اتباع استراجية ليوناردو فان المساعد سيرتاع اكثر عدد من المرات.

مثال 1

افرض انه N =4 وهكذا فانه عندنا 4 الوان مرقمة من 0 الى 3 وعندنا اربع طلبات افترض انه عدد مرات الطلبات C = (2, 0, 3, 0). وايضا افترض انه K =2 اي انه ليوناردو عنده سقاله تحتمل لونين في اي وقت وكما هو مذكور اعلاه فان السقالة تحتوي على الالوان من 0 الى 1 وسنكتب محتوى الالوان الموجودة على السقالة على النحو التالي [0, 1] . والطريقة الممكنة التي يستطيع

المساعد من خلالها تلبية طلبات الالوان على النحو التالي

الطلب الاول للون رقم 2 غير موجود على السقالة فالمساعد يضعه على السقالة ويقرر تنزيل اللون رقم 1 الى المستودع والاوان الموجودة على الساقلة هي [0, 2].

الطلب الثالي للون هو للون رقم 0فهو موجود على السقالة وهكذا المساعد سيرتاح

للطلب الثالث (عدد 3)، المساعد يزيل اللون 0، تغيير سقالة ل[3، 2].

وأخيرا، ولون آخر طلب (عدد 0) يجب أن يؤخذ من الرف إلى السقالة. المساعد قرر إزالة اللون 2، ويصبح الآن سقالة [3، 0].

لاحظ أنه في المثال أعلاه المساعد لم يتبع استراتيجية ليوناردو المثلى. الاستراتيجية المثلى من شأنها أن تزيل اللون 2 في الخطوة الثالثة، وبالتالي فإن المساعد يمكن الراحة مرة أخرى في الخطوة الأخيرة.

استراتيجية المساعد عندما تكون ذاكرته محدودة

في الصباح، المساعد يسال ليوناردو لكتابة C على قطعة من الورق، حتى يتمكن من إيجاد واتباع الاستراتيجية المثلى. ومع ذلك، يريد ليوناردة ان يحتفظ بتقنيات عمله السريه ، لذلك فهو يرفض السماح للمساعد الحصول على الورقة. انه يسمح للمساعد فقط قراءة C وتذكرها.

للأسف، ذاكرة مساعد سيئة للغاية. هو فقط قادر على تذكر بحجم M bits. بشكل عام، وهذا قد منعه من أن يكون قادر على إعادة بناء كامل تسلسل C وبالتالي، فإن على المساعد ان يأتي بطريقة ذكية لحساب تسلسل من البتات يمكنه تذكرها. ونحن نسمي هذا التسلسل بتسلسل (النصيحة) وسنرمز لها بالرمز A.

مثال 2

في الصباح، يمكن للمساعد أن ياخد ورقة ليوناردو مع تسلسل C، وقراءة التسلسل، وجعل جميع الخيارات اللازمة. وشيء واحد يمكنه اختياره وهو اختبارحالة السقالة بعد كل طلب. على سبيل المثال، عند استخدام استراتيجية (دون المستوى الأمثل) الواردة في مثال 1، فإن تسلسل حالات السقالة تكون [0، 2]، [0، 2]، [3، 2]، [3، 0]. وتذكر انه يعلم أنه الحالة الأولية للسقالة هي [0، 1].

نفترض أن لدينا M = 16، وبالتالي فإن المساعد قادر على تذكر ما يصل إلى 16 بت من المعلومات. كما ان N = 4، يمكن أن نقوم بتخزين كل لون باستخدام 2 بت. ولذلك 16 بت تكفي لتخزين تسلسل حالات السقالةالمذكورة أعلاه. وبالتالي يحسب المساعد تسلسل النصيحة التالية: A = (0، 0، 1، 0، 0، 0، 1، 0، 1، 1، 1، 0، 1، 1، 0، 0).

في وقت لاحق من نفس اليوم، يمكن للمساعد فك هذا التسلسل (تسلسل النصيحة)واستخدامها لتحديد خياراته.

و بطبيعة الحال، مع M = 16 يمكن أيضا للمساعد اختيار تذكر تسلسل كامل C بدلا من ذلك، مستخدما في ذلك فقط 8 من 16 بت المتوفرة. في هذا المثال نرغب فقط لتوضيح أنه قد يكون خيارات أخرى، دون التخلي عن أي الحل جيد. بيان

Statement

يتوجب عليك كتابة برنامجين منفصلين منفصلين في لغة البرمجة نفسها. وسيتم تنفيذ هذه البرامج بشكل تسلسلي، دون أن يتمكن من التواصل مع بعضهم البعض أثناء تنفيذ.

وسوف يكون أول برنامج هوالمستخدم من قبل المساعد في الصباح. وستعطى هذا البرنامج التسلسل C ، وdj,[f حساب تسلسل النصيحة A.

والبرنامج الثاني هو المستخدم من قبل المساعد أثناء النهار. سيتلقى هذا البرنامج تسلسل النصيحة A، ومن ثم لديها لمعالجة

التسلسل C من طلبات ليوناردو. سوف نلاحظ أن تسلسل C فقط يتم الكشف عنها لهذا البرنامج بمعدل طلب واحد في كل مرة، وكل طلب لابد من معالجته قبل استقبال الطلب التالي.

على نحو أدق، في أول برنامج يتوجب عليك تنفيذ ComputeAdvice(C, N, K, M)وادخال مجموعة من الأعداد C من N الاعداد الصحيحة(0، ...، N - 1)، وعدد من الألوان K على السقالة، وM عدد البتات المتاحة للحصول على تسلسل النصيحة. يجب على هذا البرنامج حساب تسلسل النصيحة A والذي يتكون من M بت. يجب أن برنامج التواصل مع التسلسل A إلى النظام عن طريق استدعاء كل بت من A على تسلسل الروتين التالي:

WriteAdvice(B) — append the bit B to the current advice sequence A. (You can call this routine at most M times.)

في البرنامج الثاني عليك تنفيذ برنامج واحد Assist(A, N, K, R). المدخل إلى هذا البرنامج هو تسلسل النصيحة A ,الأعداد الصحيحة Nو K كما هو محدد أعلاه، و طول الفعلي R للتسلسل النصيحة A في بت ($R \leq M$). يجب على هذا البرنامج تنفيذ استراتيجيتك المقترحة لمساعد، وذلك باستخدام نامج التالي:

GetRequest() — returns the next color requested by Leonardo. (No information about the future requests is revealed.)

PutBack(T) — put the color T from the scaffold back to the shelf. You may only call this routine with T being one of the colors currently on the scaffold.

عند التنفيذ، يجب أن يساعد برنامجك Assist استدعاء GetRequest بالضبط N مرات، في كل مرة تتلقي احد طلبات ليوناردو بالترتيب، في النظام. بعد كل استدعاء GetRequest، إذا كان لون عودتها ليست في السقالة، يجب أيضا استدعاء PutBack (T) مع اختيارك من T. وإلا، يجب أن لا ندعو PutBack. ويعتبر عدم القيام بذلك خطأ وأنه سوف يتسبب في إنهاء البرنامج. يرجى نتذكر انه في بداية السقالة يحتوي على ألوان من 0 إلى K - 1، ضمنا.

حالة اختبار معينة يمكن اعتبارها محلولة إذا كان برنامجيك تتبع متابعة جميع القيود المفروضة، والعدد الإجمالي للاستدعات إلى PutBack يساوي بالضبط إلى استراتيجية ليوناردو المثلى. ملاحظة أنه إذا كان هناك استراتيجيات متعددة التي تحقق نفس العدد من الاستدعات إلى PutBack، برنامجك مسموح له أداء أي واحد منهم. (أي أنه غير مطلوب لمتابعة استراتيجية ليوناردو، إذا كان هناك استراتيجية أخرى جيدة على حد سواء.)

مثال 3

استمرار المثال 2 نفترض أن كنت في ComputeAdvice محسوب A = (0، 0، 1، 0، 0، 0، 1، 0، 1، 1، 1، 0، 1، 1، 0، 0، 0، 1، 0، 0). من أجل التواصل إلى النظام، سيكون لديك لجعل التسلسل التالي من الاستدعات: WriteAdvice (0)، WriteAdvice (0)، WriteAdvice (1)، WriteAdvice (0)، WriteAdvice (0)، WriteAdvice (0)، WriteAdvice (1)، WriteAdvice (0)، WriteAdvice (1)، WriteAdvice (1)، WriteAdvice (0)، WriteAdvice (1)، WriteAdvice (1)، WriteAdvice (0)، WriteAdvice (0)

وثم المساعدة روتينك الثانية يمكن تنفيذها، وتلقي تسلسل أعلاه A، والقيم K = 2، N = 4، و R. = 16 والمساعد له ثم الروتينية لأداء بالضبط N = 4 دعوات لGetRequest. أيضا، بعد بعض تلك الطلبات، سوف يساعد علي أن أتصل PutBack (T) مع اختيار مناسبة من T.

ويبين الجدول أدناه سلسلة من الاستدعاتالتي يتوافق مع الخيارات (دون المستوى الأمثل) من مثال 1. الواصلة يدل على عدم وجود مكالمة لPutBack.

العمل	GetRequest()
PutBack(1)	2
-	0
PutBack(0)	3
PutBack(2)	0

البرنامج الفرعي 1 8(نقاط)

- $N \leq 5\ 000$.

يمكنك استخدام على الأكثر $M = 65\ 000$ bits.

البرنامج الفرعي 2 (9 نقاط)

- $N \leq 100\ 000$.

يمكنك استخدام على الأكثر $M = 2\ 800\ 000$ bits.

البرنامج الفرعي 3 (9 نقاط)

- $N \leq 100\ 000$.
- $K \leq 25\ 000$.

يمكنك استخدام على الأكثر $M = 1\ 500\ 000$ bits.

البرنامج الفرعي 4 (35 نقطة)

- $N \leq 5\ 000$.

يمكنك استخدام على الأكثر $M = 10\ 000$ bits.

البرنامج الفرعي 5 (39 نقطة)

- $N \leq 100\ 000$.
- $K \leq 25\ 000$.

يمكنك استخدام على الأكثر $M = 1\ 800\ 000$ bits.

The score for this subtask depends on the length R of the advice your program communicates. More precisely, if R_{max} is the maximum (over all test cases) of the length of the advice sequence produced by your routine `ComputeAdvice`, your score will be:

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

Implementation details

.You should submit exactly two files *in the same programming language*

The first file is called `advisor.c`, `advisor.cpp` or `advisor.pas`. This file must implement the routine `ComputeAdvice` as described above and can call the routine `WriteAdvice`. The second file is called `assistant.c`, `assistant.cpp` or `assistant.pas`. This file must implement the routine `Assist` as described above and can .call the routines `GetRequest` and `PutBack`

.The signatures for all the routines follow

C/C++ programs

```
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();
```

Pascal programs

```
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;
```

These routines must behave as described above. Of course you are free to implement other routines for their internal use. For C/C++ programs, your internal routines should be declared `static`, as the sample grader will link them together. Alternately, just avoid having two routines (one in each program) with the same name. Your submissions must not interact in any way with standard .input/output, nor with any other file

When programming your solution, you also have to take care of the following instructions (the .(templates you can find in your contest environment already satisfy the requirements listed below

C/C++ programs

At the beginning of your solution, you have to include the file `advisor.h` and `assistant.h`, respectively, in the advisor and in the assistant. This is done by including in your :source the line

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

or

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

The two files `advisor.h` and `assistant.h` will be provided to you in a directory inside your contest environment and will also be offered by the contest Web interface. You will also be provided (through the same channels) with code and scripts to compile and test your solution. Specifically, after copying your solution into the directory with these scripts, you will have to run (`compile_c.sh` or `compile_cpp.sh` (depending on the language of your code.

Pascal programs

You have to use the units `advisorlib` and `assistantlib`, respectively, in the advisor and in the assistant. This is done by including in your source the line:

```
uses advisorlib;
```

or

```
uses assistantlib;
```

The two files `advisorlib.pas` and `assistantlib.pas` will be provided to you in a directory inside your contest environment and will also be offered by the contest Web interface. You'll also be provided (through the same channels) with code and scripts to compile and test your solution. Specifically, after copying your solution into the directory with these scripts, you will have to run `compile_pas.sh`.

Sample grader

The sample grader will accept input formatted as follows:

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

The grader will first execute the routine `ComputeAdvice`. This will generate a file `advice.txt`, containing the individual bits of the advice sequence, separated by spaces and terminated by a 2.

Then it will proceed to execute your `Assist` routine, and generate output in which each line is either of the form "`R [number]`", or of the form "`P [number]`". Lines of the first type indicate calls to `GetRequest()` and the replies received. Lines of the second type represent calls to `PutBack`() and the colors chosen to put back. The output is terminated by a line of the form "`E`".

Please note that on the official grader the running time of your program may differ slightly from the time on your local computer. This difference should not be significant. Still, you are invited to use the test interface in order to verify whether your solution runs within the time limit.

Time and Memory limits

- .Time limit: 7 seconds
- .Memory limit: 256 MiB