Asia-Pacific Informatics Olympiad
アジア太平洋情報オリンピック
2008年5月10日

課題	BEADS	ROADS	DNA
入力	標準入力		
出力	応答	標準出力	
時間制限	2 秒	1 秒	1 秒
メモリ制限	256 MB	128 MB	128 MB
配点	100	100	100
	300		

言語	コンパイルオプション
C	`gcc -o abc abc.c -std=c99 -O2 -DCONTEST -s -static -lm`
C++	`g++ -o abc abc.cpp -O2 -DCONTEST -s -static`
Pascal	`fpc -O1 -XS -dCONTEST abc.pas`

試験時間: 5 時間

試験問題数: 3 問（全問必答）

日本語／Japanese

ビーズ： Beads

X 教授は最近，彼の最新の発明である「究極のビーズ交換機 (the Ultimate Bead Swapper，以下 UBS)」を世界に発表した．名前の通り，それを使うと，列に含まれるビーズの場所を交換することができて，ビーズの列をいっそう面白くできるのだ！

UBS には南北の方向に平行に設置された N 個の**ベルトコンベヤー** (conveyer belt) がある．ベルトコンベヤーは左から右に順番に 1 から N まで番号がつけられている．全てのベルトは北から南の方向へ同じスピードで動いている．また，隣接するコンベヤーの間に M 個の**交換機** (swapper) が設置されている．どの 2 つの交換機も UBS の北端からの距離は異なっている．(つまり，交換機は UBS の北端からの距離で完全に順序付けることができる.) 交換機は北にあるものから順に 1 から M まで番号がつけられている．図 1 は上から見たときの UBS の図である．

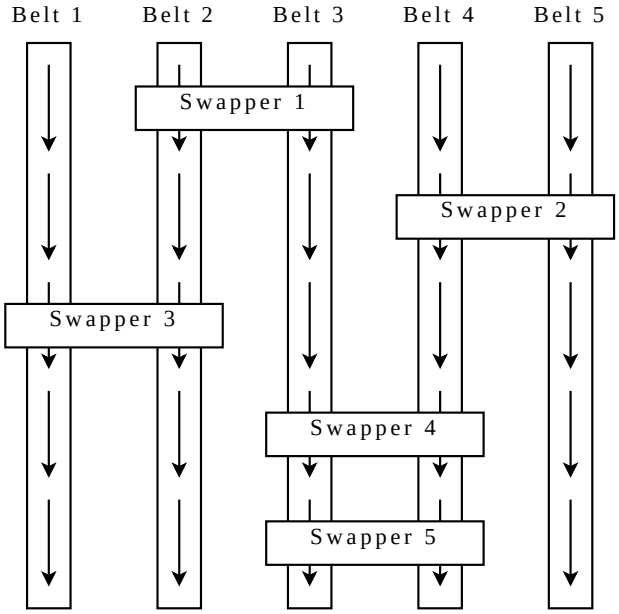

図 1: 5 個のベルトコンベヤーと 5 個の交換機がある UBS

UBS を使うには，N 個のビーズをベルトコンベヤーの北端に同時に置き，ベルトに乗って水平に 1 列に並んで動くようにする．2 つのビーズが交換機の下に来たとき，右のベルトコンベヤー上のビーズは左のベルトコンベヤーへ移動され，左のベルトコンベヤー上のビーズは右のベルトコンベヤーへと移動される．交換の際，水平の 1 列の並びは崩れない．図 2 は交換機の動作を表している．

課題

ベルトコンベヤーの数 N と交換機の数 M，それぞれの交換機の位置が与えられたとき，以下の形式の質問に答えるプログラムを作成せよ．

> K と J が与えられた時，UBS の北端でベルトコンベヤー K に置かれたビーズが，ビーズの列が交換機 J を通過した直後にどのベルトコンベヤーに乗っているか．

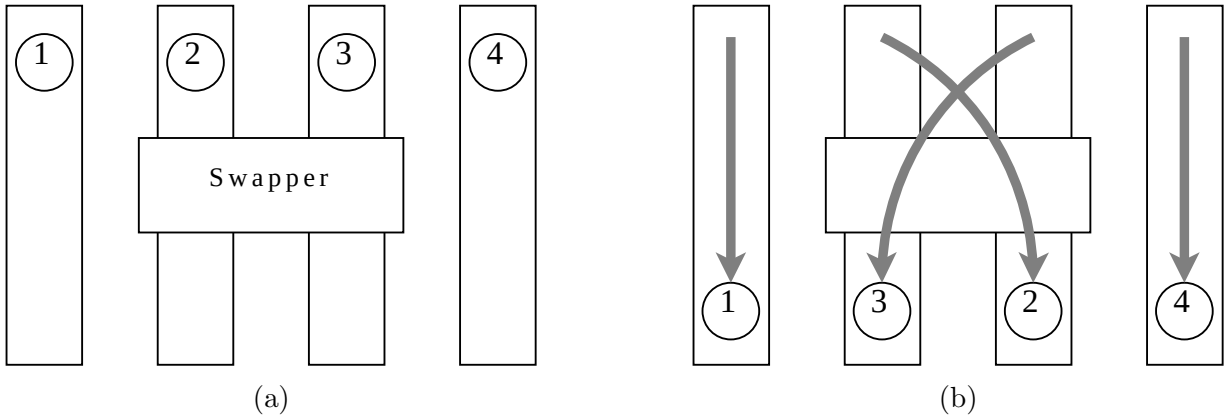

図 2: (a) 4 つのビーズがベルトコンベヤーに沿って動く様子 (b) ビーズ 2 と 3 が交換機の下を通り場所が交換される様子

入力

標準入力から入力せよ．最初の行はベルトコンベヤーの数を表す整数 $N(1 \leqq N \leqq 300,000)$ と交換機の数を表す整数 $M(1 \leqq M \leqq 300,000)$ を含む．

続く M 行はそれぞれ交換機の位置を表している．交換機の位置は北端に近いものから順番に与えられる．それぞれの行は 1 つの整数 $P(1 \leqq P \leqq M-1)$ を含み，P はその行に対応する交換機がベルトコンベヤー P と $P+1$ の上に設置されていることを意味する．

対話

上記の入力の読み取りが完了した後，表 1 にあるライブラリの関数を呼び出せ．関数は以下の順番に呼び出さなければいけない．

1. 関数 `getNumQuestions` を呼び出し，尋ねられる質問の個数 $Q(1 \leqq Q \leqq 300,000)$ を取得する．
2. 以下を Q 回繰り返す．
   (a) 関数 `getQuestion` を呼び出し，質問を 1 つ取得する．
   (b) 関数 `answer` を呼び出し，取得した質問に対して解答をする．

`getNumQuestions` を最初に呼び出さなければならないこと，またその 1 回しか呼び出してはならないことを強調しておく．`getQuestion` と `answer` は交互に呼び出さなければならない．つまり，`getQuestion` を 1 回呼び出した後，`answer` の呼び出しを行うまでは `getQuestion` を呼び出してはいけない．逆もまた同様である．プログラムがこれら取り決めを守らない動作をした場合，あなたのそのシナリオ についての点数は 0%となる．

プログラムする際の指示

Pascal のソースコードを提出するのであれば，以下の記述を含まなければいけない．

```
uses beadslib;
```

C か C++のソースコードを提出するのであれば，以下の 1 行を含まなければいけない．

```
#include "beadslib.h"
```

関数のプロトタイプ	説明

Pascal
`function getNumQuestions():longint`
C and C++
`int getNumQuestions()`
	尋ねられる質問の個数を返す．

Pascal
`procedure getQuestion(var K:longint, var J:longint)`
C
`void getQuestion(int *K, int *J)`
C++
`void getQuestion(int &K, int &J)`
	`K` には UBS の北端でビーズが置かれるベルトコンベヤーの番号が与えられる．
	`J` には交換機の番号が与えられる．

Pascal
`procedure answer(x:longint)`
C and C++
`void answer(int x)`
	直前の `getQuestion` の呼び出しで取得した質問に対応する解答は `x` であると報告する．

表 1: 対話ライブラリ

ライブラリとプログラムのサンプル

ライブラリとプログラムのサンプルのソースコードを含む zip ファイルが提供される．ファイルには `pascal`，`c`，`cpp` の 3 つのディレクトリが含まれ，それぞれ Pascal，C，C++のソースコードである．それぞれのディレクトリには，対話ライブラリのサンプルのソースコードと，ライブラリの関数を正しい順番で呼び出すプログラムのソースコードが含まれている．

Pascal については，サンプルの対話ライブラリはユニット `beadslib` に含まれており，ソースコードは `beadslib.pas` として与えられている．ファイル `sample.pas` はライブラリを正しく使うプログラムのソースコードである．

C については，ライブラリ関数のプロトタイプ宣言が `beadslib.h` で与えられている．関数は `beadslib.c` 内に記述されている．ファイル `sample.c` はライブラリを正しく使うプログラムのソースコードである．

C++についても，ライブラリ関数のプロトタイプ宣言が `beadslib.h` で与えられている（内容は C 用のものと異なる）．関数は `beadslib.cpp` 内に記述されている．ファイル `sample.cpp` はライブラリを正しく使うプログラムのソースコードである．

サンプルのライブラリは以下のように動作する．

- `getNumQuestions` が呼び出されると，ファイル `questions.txt` を開き，質問の個数を読み取り返す．
- `getQuestion` が呼び出されると，`questions.txt` から `K` と `J` を読み取る．
- `answer` が呼び出されると，引数 `x` を標準出力に表示する．
- 誤った順番で関数が呼ばれる度に，ライブラリはエラーメッセージを標準出力に表示する．

ファイル `questions.txt` のフォーマットは以下の通りである．1 行目には質問の個数 Q を含む．続く Q 行は，それぞれ 2 個の整数 K，J を含む．K はベルトコンベヤーの番号であり，J は交換機の番号である．

入力例

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

`questions.txt` の例

```
2
3 4
5 5
```

(この入力例は図 1 に対応している．)

対話例

関数呼び出し	関数からの返り値や説明
`getNumQuestions();`	2 プログラムはこれから 2 個の質問を受ける．

Pascal
`getQuestion(K, J);`
C
`getQuestion(&K, &J);`
C++
`getQuestion(K, J);`
	`K=3, J=4`
	UBS の北端でベルトコンベヤー 3 に置かれたビーズが，ビーズの列が交換機 4 を通過した直後にどのベルト上に乗っているか？

`answer(1);`	そのビーズは，ビーズの列が交換機 4 を通過した直後にベルトコンベヤー 1 上にある．

Pascal
`getQuestion(K, J);`
C
`getQuestion(&K, &J);`
C++
`getQuestion(K, J);`
	`K=5, J=5`
	UBS の北端でベルトコンベヤー 5 に置かれたビーズが，ビーズの列が交換機 5 を通過した直後にどのベルト上に乗っているか？

`answer(4);`	そのビーズは，ビーズの列が交換機 5 を通過した直後にベルトコンベヤー 4 上にある．

時間とメモリの制限

あなたのプログラムは 2 秒以内に停止しなければならず，256 MB よりも多くのメモリを使用してはならない.

採点基準

各シナリオについて，プログラムが関数呼び出しの取り決めを守り，全ての質問に対し正しく解答した場合は 100%の得点が得られ，それ以外は 0%となる.

配点の 20 点分のシナリオは，$M, Q \leqq 10,000$ を満たす.

道路：Roads

New Asia **王国**には N 個の村があり，M 本の道路がそれらを結んでいる。おのおのの道路は，敷石かコンクリートで舗装されている．道路を無料にしておくのには多くのお金がかかり，この王国が全ての道路を保守するのは不可能に思われるので，新しい道路保守計画を立案しなければならなくなった．

国王は，王国のどの 2 つの村も無料の道路だけからなる経路で結ばれており，しかも，そのような経路は**ちょうど** *1* **本**となるようにした上で，無料の道路をできる限り減らすことを決めた．また，現代の交通機関にはコンクリートの道路のほうが適しているが，国王は敷石の道路を歩くのも風流であると思っている．そこで, 国王は敷石の道路のうち**ちょうど** K 本を無料のままにすると決めた．

例えば，New Asia 王国には図 1a のように村と道路があるとする．国王が 2 本の敷石の道路を無料にしたいと望んでいるとすると，王国は図 1b のように (1,2), (2,3), (3,4), (3,5) の道路を無料のままにできる．この計画は次の 3 つのことから国王の要求を満たしている．

(1) どの 2 つの村も無料の道路だけからなる経路で結ばれており，そのような経路はちょうど 1 本である．

(2) 無料の道路の数はできる限り少なくなっている．

(3) 無料の道路にちょうど 2 本の敷石の道路 (2,3), (3,4) が含まれている．

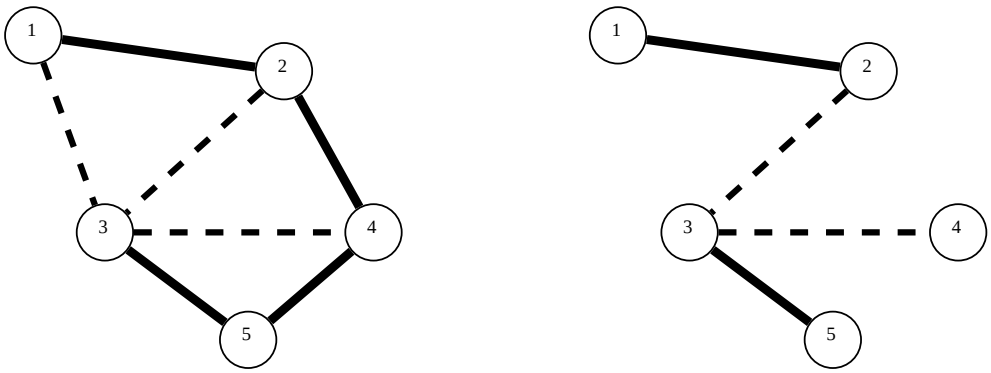

図 1: (a) New Asia 王国の村と道路の状況の例．実線がコンクリートの道路を，点線が敷石の道路を表している．(b) 2 本の敷石の道路を無料にする道路保守計画．無料の道路だけが記載されている．

課題

あなたの課題は，New Asia 王国の記述と無料のままにしたいと国王が望む敷石道路の本数が与えられると，王の意向に沿う道路保守計画が可能かどうかを調べ，もしそのような計画が可能な場合はそれらのうちの 1 つを出力するプログラムを書くことである．

入力

1 行目には，3 つの整数が空白区切りで書かれている:

- N, 村の数 ($1 \leqq N \leqq 20,000$)
- M, 道路の数 ($1 \leqq M \leqq 100,000$)
- K, 無料のままにしたいと国王が望んでいる敷石道路の本数 ($0 \leqq K \leqq N-1$)

続く M 行には，New Asia 王国の道路の情報が書かれており，道路には 1 から M の番号がついている．$(i+1)$ 行目には，道路 i の情報を表す 3 つの整数が空白区切りで書かれている:

- u_i と v_i, 村 u_i と 村 v_i は道路 i が結ぶ村であり，村には 1 から N の番号がついている
- c_i, 道路 i のタイプ; 道路 i が敷石の道路なら $c_i = 0$ で，コンクリートの道路なら $c_i = 1$

どの 2 つの村の間にも高々1 本の道路しかない.

出力

もし国王の要求を満たす道路保守計画が存在しない場合は，あなたのプログラムは最初の行に `no solution` を出力しなさい.

その他の場合は，無料のままにする道路を列挙することで, 条件を満たす道路保守計画を出力しなさい. 道路を列挙する際は，1 行に 1 つの道路を書くこと．また，入力中のそれぞれの道路の記述をそのまま出力すること．その際に道路を出力する順序は問わない．もし複数の解がある場合は，どの解を出力しても良い.

入力例

```
5 7 2
1 3 0
4 5 1
3 2 0
5 3 1
4 3 0
1 2 1
4 2 1
```

(この入力は図 1a に対応している．)

出力例

```
3 2 0
4 3 0
1 2 1
5 3 1
```

(この出力は図 1b に対応している．)

時間とメモリの制限

あなたのプログラムは 1 秒以内に停止しなければならず，128 MB よりも多くのメモリを使用してはならない.

採点基準

各入力に対して，正しい解答を出力した場合は 100% の得点が得られ，それ以外は 0% となる．
入力の 20 点分において，K は高々 10 である．

DNA

興味深いコンピュータの使用方法の一つとして，DNA 配列などの生物学的データの分析がある．生物学的に，DNA 鎖は，アデニン (Adenine)，シトシン (Cytosine)，グアニン (Guanine)，チミン (Thymine) からなるヌクレオチドが鎖のように連なっている．これら 4 つのヌクレオチドは それぞれ `A`，`C`，`G`，`T` の文字として表される．従って，DNA 鎖は，それら 4 つの文字からなる文字列として表すことができる．我々は，そのような文字列を *DNA* **配列**と呼ぶ．

生物学者は，DNA 鎖の中のいくつかのヌクレオチドを決定することができない可能性がある．そのような場合，DNA 鎖配列の中の不明なヌクレオチドを表すための文字として，`N` が使われる．言い換えれば，`N` は，`A`，`C`，`G`，`T` の内のいずれか一文字を表すワイルドカード文字である．我々は，`N` を一文字以上含む DNA 配列を，**不完全配列**と呼ぶ．一方，`N` を一文字も含まない DNA 配列を，**完全配列**と呼ぶ．また，不完全配列の各 `N` に，4 つのヌクレオチドの内の 1 つを代入した完全配列を，その不完全配列に**整合する**と呼ぶ．例えば，`ACCCT` は，`ACNNT` に整合する．しかし，`AGGAT` は，`ACNNT` には整合しない．

研究者は，大抵 4 つのヌクレオチドを，アルファベット順に並べる．つまり，`A` は `C` より前に現れ，`C` は `G` より前に現れ，`G` は `T` より前に現れる．DNA 配列の全てのヌクレオチドが，直に右に連なるヌクレオチドに対して，アルファベット順で同じか前である DNA 配列を，**型**-1 と分類する．例えば，`AACCGT` は型-1 である．しかし，`AACGTC` は型-1 ではない．

一般に，型-j ($j > 1$) であるとは，型-$(j-1)$ であるか，または型-$(j-1)$ と型-1 の連結の場合である．例えば，`AACCC`，`ACACC`，`ACACA` は，型-3 である．しかし，`GCACAC`，`ACACACA` は，型-3 ではない．

さらに，研究者は，DNA 配列を辞書式順序に並べる．この順序における型-3 の長さ 5 の配列は，最初が `AAAAA` であり，最後が `TTTTT` である．別の例として，不完全配列 `ACANNCNNG` に整合する型-3 の配列の辞書式順序の最初の 7 つを以下に挙げる．

```
ACAAACAAG
ACAAACACG
ACAAACAGG
ACAAACCAG
ACAAACCCG
ACAAACCGG
ACAAACCTG
```

課題

与えられた長さ M の不完全配列に整合する R 番目の型-K の配列を出力するプログラムを作成せよ．

入力

1 行目には 3 つの整数 $M(1 \leqq M \leqq 50,000)$，$K(1 \leqq K \leqq 10)$，$R$ $(1 \leqq R \leqq 2 \times 10^{12})$ が空白区切りで書かれている．2 行目には，長さ M の文字列として，不完全配列が書かれている．不完全配列に整合する型-K の配列数は，4×10^{18} を超えない．つまり，C と C++を用いる場合は `long long` 型を使う

ことで，Pascal を用いる場合は Int64 型を使うことで，この数を表すことができる．また，R は与えられた不完全配列に整合する型-K の配列数を超えない．

出力

一行に，入力で与えられた不完全配列に整合する辞書式順序で R 番目の型-K の配列を出力せよ．

入力例 1

```
9 3 5
ACANNCNNG
```

出力例 1

```
ACAAACCCG
```

入力例 2

```
5 4 10
ACANN
```

出力例 2

```
ACAGC
```

プログラミングする際の注意

C または C++を用いる場合，long long 型を使わなければならない．以下に long long 型を標準入力から読み込んだり標準出力に書き出す例を示す．

```
long long a;
scanf("%lld",&a);
printf("%lld\n",a);
```

Pascal を用いる場合，Int64 型を使わなければならない．このデータを扱う際に特別な命令を使う必要はない．

時間とメモリの制限

あなたのプログラムは 1 秒以内に停止しなければならず，128 MB よりも多くのメモリを使用してはならない．

採点基準

各入力に対して，正しい解答を出力した場合は 100% の得点が得られ，それ以外は 0% となる．
入力の 20 点分において，M は高々 10 である．