Host Country Report

Asia Pacific Informatics Olympiad (APIO2008)

Backgrounds

The Asia-Pacific Informatics Olympiad (APIO) is an online contest for the South Asian / Western Pacific region. The format contest is that students do the exam at a proctored site in their countries and submit their works to the host countries via Internet. Awards are announced at the APIO meeting at the IOI venue in the same year.

The first APIO was hosted by Australia. The host of the APIO 2008 is Thailand. This event is sponsored by Institute for promotion of teaching science and technology, Thailand. The participating countries are Australia, China, Chinese Taipei, India (unofficial), Indonesia, Japan, Korea, Macao, New Zealand, Philippines and Sri Lanka. The contest is scheduled on May 10, 2008.

APIO Competition Tasks

APIO tasks are basically the same as tasks in International Olympiad in Informatics, which are typically focused on the design of efficient, correct algorithms. Input and output are to be kept as simple as possible. The guidelines of task preparations of IOI should be considered, with no restriction. The documents about submission of IOI2008 tasks, which can be used as reference to create the tasks, can be found at http://ioi.ms.mff.cuni.cz/sc/documents/.

Awarding policy

- For each team, the top six scorers are the official team. If there are ties at the 6th, all contestants who tie at 6th are included into the official team.
- Gold medal awarding must be decided first. Seven highest-scorers will be awarded gold medals. However, if there are ties at the 7th, all who tie at the 7th will be awarded gold medals. After that, silvers and bronzes are decided, respectively, by the same way as the gold medal. However, the total number of medals is 42 except that there are tie at the 42nd.
- Scores of unofficial team are open to head of delegation of each country.

Notes for improvements

- Since task is the most important part of the competition, it would be nice if each participating country must submit a task. It may like an application form to join the contest. If they cannot do so, they should acknowledge the host whether they want to participate or not. It is not good to leak the task script to a person who did not join the event.
- The task is quite hard for average students and quite easy for top students. As shown in results, there are 25 full-scorers; mean is 108; and median is 128. If the task is still like this, it cannot distinguish the best and it discourages the average. We suggested that there should be two leagues running concurrently: tough and moderate competitions. Tough competition is 5 hours long and moderate competition should be 4 hours long. The moderate competition should start one hour behind the tough competition. All students must attend competition of tough tasks. After the first hour, they can decide to switch to moderate competition. This should avoid bimodal distributions of full and zero scores.

Credits

APIO2008 Host

Sponsor

Institute for Promotion of teaching Science and Technology (IPST)

Advisory Committee

Sub-committee on organizing Informatics Olympiad, IPST

Organizing Committee

Punpiti Piamsa-nga (Chair)	Kasetsart University
Jittat Fakcharoenphol	Kasetsart University
Cheeraporn Sankawetai	Institute for promotion of teaching science and technology

Scientific Committee

Jittat Fakcharoenphol (Chair)	Kasetsart University
Chaiporn Jaikaeo	Kasetsart University
Pramook Khungurn	Kasetsart University
Taweesak Kijkanjanarat	Thamasat University
Bundit Lekanukit	Kasetsart University
Punpiti Piamsa-nga	Kasetsart University
Thanawin Rakthanmanon	Kasetsart University
Sitichai Sri-on	Kasetsart University
Attasit Surarerk	Chulalongkorn University

Technical Committee

Jittat Fakcharoenphol (Chair)	Kasetsart University
Pramook Khungurn (Grader)	Kasetsart University
Bundit Lekanukit (Testing)	Kasetsart University
Thanawin Rakthanmanon (Testing)	Kasetsart University

Australia

Delegation leader

Benjamin Burton (Leader)
Bernard Blackham (Deputy Leader)

Site managers and proctors

Jonathan Kummerfeld
Patrick Coleman
Clarice McLean

Coaches

Benjamin Burton (Head)
Bernard Blackham

Bangladesh

Delegation leader

Kaykobad Mohammad (Leader)	Bangladesh University of Engineering and Technology

China

Delegation leader

Hong Wang (Leader)	Tsinghua University, Beijing, China
Baolin Yin (Deputy Leader)	Bei Hang University, Beijing, China

Site managers

Guoren Wang	Northeast University, Shenyang, China
Weizu Huang	Northeast University, Shenyang, China

Site proctors

Zhi Guo	Northeast University, Shenyang, China
Meian Zhang	Northeast University, Shenyang, China
Hongjun Chen	Northeast University, Shenyang, China
Dengke Zhang	Northeast University, Shenyang, China

Head coach

Hong Wang	Tsinghua University, Beijing, China
Wenhu Wu	Tsinghua University, Beijing, China

Technical Supports

Weidong Hu	Tsinghua University, Beijing, China
Gelin Zhou	Tsinghua University, Beijing, China

System and Network Supporters

Qiyang Zhao	Bei Hang University, Beijing, China
Haifeng Ding	Bei Hang University, Beijing, China
Bin Yang	Bei Hang University, Beijing, China

Chinese Taipei

Delegation Leader

Greg Lee (Leader)	National Taiwan Normal University
Jia-Ling Koh (Deputy Leader)	National Taiwan Normal University
Berlin Chen (Deputy Leader)	National Taiwan Normal University

Site Manager

Rong-Guey Ho	National Taiwan Normal University

Site Proctor

H. C. Yo	National Taiwan Normal University

Coaches:

Rong-Guey Ho	National Taiwan Normal University
Greg Lee	National Taiwan Normal University
Jia-Ling Koh	National Taiwan Normal University
Berlin Chen	National Taiwan Normal University

India

Delegation leader

Madhavan Mukund (Delegation Leader)	Chennai Mathematical Institute
K. Narayan Kumar (Deputy Leader)	Chennai Mathematical Institute

Indonesia

Delegation Leaders

Suryana Setiawan	Universitas Indonesia

Site managers

Suryana Setiawan (Depok Site)	Universitas Indonesia

Coaches:

Inggriani Liem	Institut Teknologi Bandung
Nurokhman	Universitas Gajah Mada
Adi Mulyanto	Institut Teknologi Bandung

Japan

Delegation leader

Seiichi Tani (Delegation Leader)	Nihon University
Masao Hara (Deputy Leader)	Tokai University

Site managers:

Eikoh Chida (Ichinoseki site)	Ichinoseki National College of Technology
Yoshimitsu Kuroki (Kurume site)	Kurume National College of Technology
Seiichi Tani (Tokyo1 site)	Nihon University
Yoshiaki Oomori (Tokyo2 site)	Tokyo Tech High School of Science and Technology
Keisuke Kawanisi (Kobe site)	Nada Junior and Senior High School
Masahito Hirose (Takasago site)	Hakuryo Junior High and High School
Yutaka Hori (Hamamatsu site)	Shizuoka Prefectural Hamamatsu Technical High School
Tadakatsu Masaki (Okinawa site)	Okinawa National College of Technology

Coaches

Seiichi Tani (Head Coach)	Nihon University
Tai Fukuzawa	Tokyo Institute of Technology
Takuya Akiba	University of Tokyo
Masaki Watanabe	University of Tokyo

Korea

Delegation leaders

Dong Yoon Kim (Delegation Leader)	Ajou University
Hee Chul Kim (Deputy Leader)	Hankuk University of Foreign Studies

Head Coach:

Tae Cheon Yang	Kyungsung University

Site Managers:

Do Kyung Lee
Yong Woon Cho

Macao, China

Delegation leaders

Teng Lam (Delegation Leader)	University of Macao

New Zealand

Delegation leader

Margot Phillipps	Lynfield College, Auckland

Site Managers

Richard Lobb	University of Canterbury
Michael Dineen	Auckland University

Philippines

Delegation Leader

Rafael P. Saldana	Ateneo de Manila University

Site Manager and Proctor:

Rafael Saldana	Ateneo de Manila University
Michael Blancaflor	Ateneo de Manila University

Coaches/Advisers:

Pablo Manalastas
Rafael Saldana
Felix Muga
Allan Espinosa
Christopher Rigor
Michael Samson
Technical Staff:
Ryan Tamayo
Mark Bautista

Singapore

Delegation leaders

Tuck Choy Aaron Tan (Delegation leader)	National University of Singapore
Zhiyong Melvin Zhang (Deputy leader)	National University of Singapore

Site managers and Proctors

Sun Teck Tan	National University of Singapore

Coaches

Tuck Choy Aaron Tan (Head coach)	National University of Singapore
Zhiyong Melvin Zhang	National University of Singapore

Delegation leaders

Sifaan Zavahir (Leader)	IOI Organizing Committee, Sri Lanka
Nayana Somaratna (Deputy Leader)	IOI Organizing Committee, Sri Lanka

Site Manager

Parinda Jayasiri (Site of University of Colombo)	IOI Organizing Committee, Sri Lanka

Coaches

Gihan Wikramanayake (Head Coach)	IOI Organizing Committee, Sri Lanka
Sifaan Zavahir	IOI Organizing Committee, Sri Lanka
Nayana Somaratna	IOI Organizing Committee, Sri Lanka
Parinda Jayasiri	IOI Organizing Committee, Sri Lanka
Chethiya Abeysinghe	IOI Organizing Committee, Sri Lanka
Dumindu Pallewala.	IOI Organizing Committee, Sri Lanka

Thailand

Delegation leader

Cheeraporn Sangkawetai	Institute for promotion of teaching science and technology

Site managers

Phanomyong Kaewprachum (Bangkok Site)	Institute for promotion of teaching science and technology
Watcharee Jumpamule (Chiangmai Site)	Chiangmai University
Orasa Tetiwat (Phitsanuloke)	Naresuan Unviersity
Amnart Pohthong (Songkhla Site)	Prince of Songkhla University
Jakkarin Sukswatchon (Chonburi Site)	Burapa University
Sunan Tongsinoot (Pattani Site)	Prince of Songkhla University
Luck Charoenwatana (Ubonrachatani Site)	Ubonrachatani University
Apisake Hongwitayakorn (Nakorn Pathom Site)	Silapakorn University
Krisanadej Jaroensutasinee (Nakorn Si thammarat Site)	Walailak University
Niphon Samakkha (Nakorn Rachasima)	Rachasima Witayalai school

Results

User	Name	beads	roads	dna	Total	Medal
CN02	Shike Mei	100	100	100	300	**Gold**
CN04	Yihan Gao	100	100	100	300	**Gold**
CN10	Junxing Wang	100	100	100	300	**Gold**
CN111	Yining Wang	100	100	100	300	**Gold**
CN113	Bin Jin	100	100	100	300	**Gold**
CN127	Chiheng Xu	100	100	100	300	**Gold**
CN15	Hanjun Xiao	100	100	100	300	**Gold**
CN16	Linyun Yu	100	100	100	300	**Gold**
CN19	Yi Liu	100	100	100	300	**Gold**
CN43	Jun Qu	100	100	100	300	**Gold**
CN79	Qinxiang Cao	100	100	100	300	**Gold**
CN80	Keqiang Luo	100	100	100	300	**Gold**
CN90	Huacheng Yu	100	100	100	300	**Gold**
CN95	Huaxing Dong	100	100	100	300	**Gold**
ID01	Irvan Jahja	100	100	100	300	**Gold**
KR01	Jae Hong Kim	100	100	100	300	**Gold**
KR02	Chan Min Kim	100	100	100	300	**Gold**
KR07	Soon Il Kwon	100	100	100	300	**Gold**
SG15	Wei Quan Lim	100	100	100	300	**Gold**
TH23	Chantat Eksombatchai	100	100	100	300	**Gold**
TH29	Tana Wattanawaroon	100	100	100	300	**Gold**
TH41	Saran Lertpradit	100	100	100	300	**Gold**
TH46	Panupong Pasupat	100	100	100	300	**Gold**
TW01	Che-Yang Wu	100	100	100	300	**Gold**
TW08	Yan-De Wu	100	100	100	300	**Gold**
TH40	Sarun Gulayanon	100	80	100	280	**Silver**
TW04	Han-Jay Yang	100	100	80	280	**Silver**
JP04	Soejima Makoto	100	100	53	253	**Silver**
TH37	Visit Pattaranutaphorn	100	50	100	250	**Silver**
AU01	Jack Murray	20	100	100	220	**Silver**
KR04	Yong Hoon Lee	100	10	100	210	**Silver**
AU08	Daniel Axtens	100	0	100	200	**Silver**
SG06	Jonathan Yee	100	80	20	200	**Silver**
TW02	You-Cheng Syu	100	0	100	200	**Silver**
SG30	Jia-Han Chiam	100	0	87	187	**Silver**
JP06	Matsumoto Eiichi	50	100	20	170	**Silver**
SG10	Zhan Xiong Chin	100	50	20	170	**Silver**
JP07	Yatoh Kohsuke	100	10	47	157	**Silver**
SG19	Kah Hou Teo	100	50	7	157	**Silver**
TW12	Hao-Cheng Kao	100	30	27	157	**Bronze**
ID03	Reinardus Surya Pradhitya	100	30	20	150	**Bronze**
ID07	Listiarso Wastuargo	100	30	20	150	**Bronze**
		100	40	0	140	
		30	10	100	140	
		100	40	0	140	
		100	10	20	130	
		100	30	0	130	
		45	80	0	125	
		100	0	20	120	

User	Name	beads	roads	dna	Total	Medal
		100	0	20	120	
		100	0	20	120	
		100	20	0	120	
		95	0	20	115	
		95	20	0	115	
		0	0	100	100	
		100	0	0	100	
		100	0	0	100	
		100	0	0	100	
		100	0	0	100	
		20	30	20	70	
		20	30	20	70	
		15	30	13	58	
		20	10	27	57	
		20	10	13	43	
		20	0	20	40	
		0	0	33	33	
		20	10	0	30	
		20	0	7	27	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		0	0	20	20	
		20	0	0	20	
		20	0	0	20	
		0	0	20	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		20	0	0	20	
		0	0	0	0	
		0	0	0	0	
		0	0	0	0	
		0	0	0	0	
		0	0	0	0	
		0	0	0	0	
		0	0	0	0	
		0	0	0	0	
		0	0	0	0	
		0	0	0	0	
		0	0	0	0	
		0	0	0	0	
		0	0	0	0	

User	Name	beads	roads	dna	Total	Medal
		0	0	0	0	
		0	0	0	0	
		0	0	0	0	
Median		72.5	0	20	108	
Mean		56.76	34.2	37	128	

Notes: This table shows only scores of official team. 14 Chinese students tie at the first place. Therefore, all 14 Chinese students are included into their official team. By regulations, number of gold medalists is 7; however, there are 25 students get full scores. All those are awarded gold medalists.

English (Official)

Task Creators

BEADS	Pramook Khungurn
DNA	Jittat Fakcharoenphol
ROADS	Yuan Zhou and Gelin Zhou

Editors

Jittat Fakcharoenphol	Kasetsart University
Chaiporn Jaikaeo	Kasetsart University
Pramook Khungurn	Kasetsart University
Taweesak Kijkanjanarat	Thamasat University
Bundit Lekanukit	Kasetsart University
Punpiti Piamsa-nga	Kasetsart University
Thanawin Rakthamanon	Kasetsart University
Sitichai Sri-on	Kasetsart University
Athasit Surarerks	Chulalongkorn University

Acknowledgement

Editors would like to thank to all APIO delegations who give comments to improve the tasks.

Asia-Pacific Informatics Olympiad
10th May 2008

TASK	BEADS	ROADS	DNA
input	standard input		
output	interaction	standard output	
time limit	2 second	1 second	1 second
memory limit	256 MB	128 MB	128 MB
points	100	100	100
	300		

Language	Compiler Directive
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`

Duration: 5 hours

3 problems

All problems should be attempted.

Beads

Professor X has recently revealed his latest invention to the world: the Ultimate Bead Swapper (UBS). As the name implies, it can make a sequence of beads much more interesting by swapping some beads in it!

The UBS has N *conveyor belts* placed in north-south direction in parallel. The conveyor belts are numbered 1 to N from left to right. Every belt moves from north to south at the same speed. There are M *swappers* placed between adjacent conveyors. No two swappers are equally far from the north end of the UBS. (In other words, they can be totally ordered according to how far they are from the north end.) The swappers are numbered 1 to M from north to south. Figure 1 shows the UBS when viewed from above.

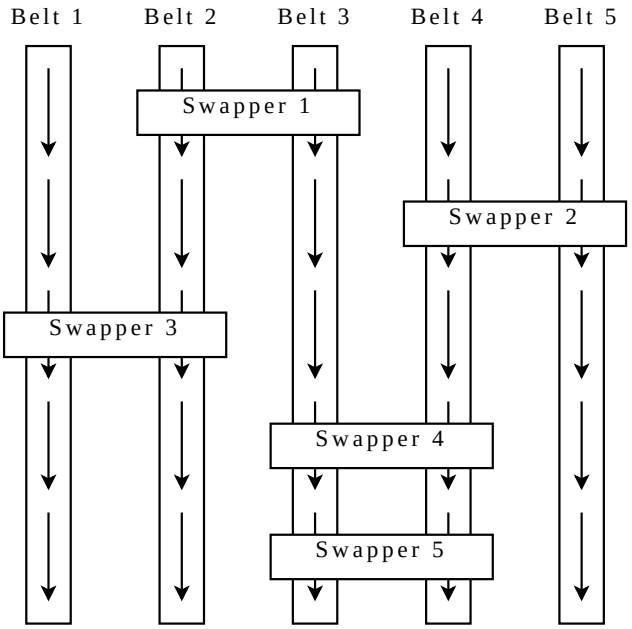

Figure 1: An Ultimate Bead Swapper with 5 conveyor belts and 5 swappers.

To use the UBS, N beads are placed at the north end of the conveyor belts at the same time so that they form a horizontal row as they move along the belt. When two beads come under a swapper, the bead on the right conveyor belt is moved to the left conveyor belt, and the bead on the left conveyor belt is moved to the right conveyor. After being swapped, the two beads do not break the horizontal row. Figure 2 illustrates the behavior of a swapper.

Task

Write a program that, given the number of conveyor belts N, the number of swappers M, and the positions of each swapper, answer questions of the form:

> Given K and J, for the bead that is placed on Belt K at the north end of the UBS, which belt is the bead on after all beads just move past Swapper J?

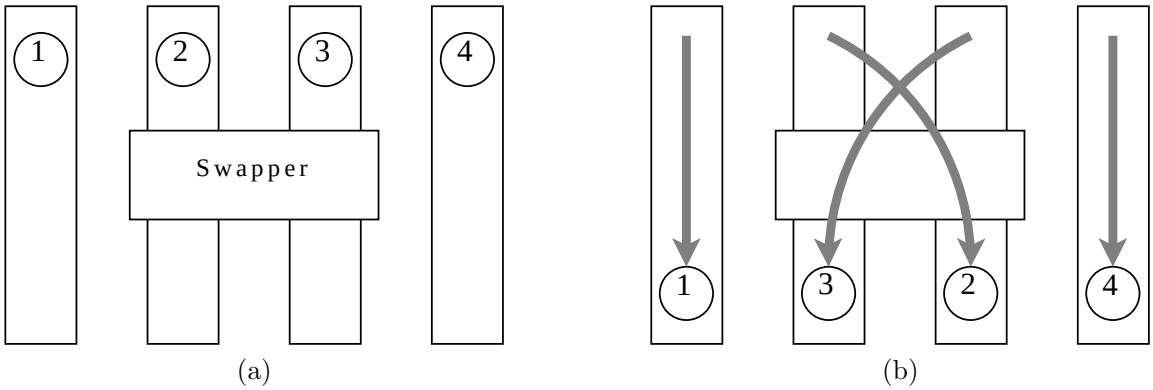

Figure 2: (a) Four beads move along the conveyor belts. (b) Bead 2 and 3 trade places after going under the swapper.

Input

Your program should read from standard input. The first line contains the number of conveyor belts $N(1 \leq N \leq 300,000)$ and the number of swappers $M(1 \leq M \leq 300,000)$.

Swappers are listed from north to south in the following M lines. Each line contains one integer $P(1 \leq P \leq M-1)$, meaning that there is a swapper over conveyor belt P and $P+1$.

Interaction

After reading the input above, your program should call functions from the following library specified in Table 1. The functions must be called in the following order:

1. It calls the function `getNumQuestions` to retrieve $Q(1 \leq Q \leq 300,000)$, the number of questions it will be asked.
2. For Q times, it should:
   (a) Call the function `getQuestion` to receive one question.
   (b) Call the function `answer` to answer the question it just received.

We emphasize that `getNumQuestions` must be called first and only once. `getQuestion` and `answer` must be called alternately: after calling one `getQuestion`, your program may not call `getQuestion` until it calls `answer`, and vice versa. If your program violates this convention when running a test scenario, you will receive a 0% score for that test scenario.

Programming Instructions

If you submit a Pascal source code, the source code must contain the statement:

```
uses beadslib;
```

If you submit a C or C++ source code, the source code must contain the line:

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

Function Prototype	Description

Pascal
`function getNumQuestions():longint`
C and C++
`int getNumQuestions()`
	Return the number of questions your program is going to be asked.

Pascal
`procedure getQuestion(var K:longint; var J:longint)`
C
`void getQuestion(int *K, int *J)`
C++
`void getQuestion(int &K, int &J)`
	`K` is set to the number of the conveyor belt that the bead is placed at the north end of the UBS.
	`J` is set to the number of the swapper.

Pascal
`procedure answer(x:longint)`
C and C++
`void answer(int x)`
	Report that the answer to the question corresponding to the last `getQuestion` is `x`.

Table 1: Interaction library.

Mock Libraries and Sample Programs

You will be provided a zip file containing source code of sample libraries and programs. The file contains three directories — `pascal`, `c`, and `cpp` — for source code in Pascal, C, and C++, respectively. Each directory will contain a source code of a mock interaction library, and the source code of a program that calls the library functions in the correct order.

For Pascal, the mock interaction library is contained in the unit `beadslib`, whose source code is given in `beadslib.pas`. The file `sample.pas` is the source code of the program that uses the library correctly.

For C, the prototypes of the mock library functions are given in `beadslib.h`. The functions are specified in `beadslib.c`. The file `sample.c` is the source code of the program that uses the library correctly.

For C++, the prototypes of the mock library functions are also given in `beadslib.h` (but the file is not the same as the one for C). The functions are specified in `beadslib.cpp`. The file `sample.cpp` is the source code of the program that uses the library correctly.

The mock library behaves as follows:

- When `getNumQuestions` of the mock library is called, it opens the file `questions.txt`, reads the number of questions, and returns what is read.
- When `getQuestion` is called, it reads `K` and `J` from `questions.txt`.
- When `answer` is called, it prints the argument `x` to standard output.
- The library prints an error message to standard output every time a function is called out of the correct order.

The file `questions.txt` has the following format. The first line contains the number of questions Q. Each of the next Q lines contains two integers K, the number of a conveyor belt, and J, the number of a swapper.

Sample Input

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

Sample Content of `questions.txt`

```
2
3 4
5 5
```

(This input agrees with Figure 1)

Sample Interaction

Function Call	Return Value(s) and Explanation
`getNumQuestions();`	`2` The program will be asked two questions.

Pascal
`getQuestion(K, J);`
C
`getQuestion(&K, &J);`
C++
`getQuestion(K, J);`
	`K=3, J=4`
	For the bead that is placed on Belt 3 at the north end of the UBS, which belt is the bead on after all beads just move past Swapper 4?

`answer(1);`	After every bead passes Swapper 4, the bead in question is on Belt 1.

Pascal
`getQuestion(K, J);`
C
`getQuestion(&K, &J);`
C++
`getQuestion(K, J);`
	`K=5, J=5`
	For the bead that is placed on Belt 5 at the north end of the UBS, which belt is the bead on after all beads just move past Swapper 5?

`answer(4);`	After every bead passes Swapper 5, the bead in question is on Belt 4.

Time and Memory Limits

Your program must terminate in 2 seconds and use no more than 256 MB of memory.

Scoring

The score for each input scenario will be 100% if your program follows the function calling convention above and answers every question correctly, and 0% otherwise.

In test scenarios worthing 20 points, M, and Q will be at most $10,000$.

DNA

One interesting use of computer is to analyze biological data such as DNA sequences. Biologically, a strand of DNA is a chain of nucleotides Adenine, Cytosine, Guanine, and Thymine. The four nucleotides are represented by characters `A`, `C`, `G`, and `T`, respectively. Thus, a strand of DNA can be represented by a string of these four characters. We call such a string a *DNA sequence*.

It is possible that the biologists cannot determine some nucleotides in a DNA strand. In such a case, the character `N` is used to represent an unknown nucleotides in the DNA sequence of the strand. In other words, `N` is a wildcard character for any one character among `A`, `C`, `G` or `T`. We call a DNA sequence with one or more character `N` an *incomplete sequence*; otherwise, it is called a *complete sequence*. A complete sequence is said to *agree with* an incomplete sequence if it is a result of substituting each `N` in the incomplete sequence with one of the four nucleotides. For example, `ACCCT` agrees with `ACNNT`, but `AGGAT` does not.

Researchers often order the four nucleotides the way we order the English alphabets: `A` comes before `C`, `C` comes before `G`, `G` comes before `T`. A DNA sequence is classified as *form*-1 if every nucleotide in it is the same as or comes before the nucleotides immediately to its right. For example, `AACCGT` is form-1, but `AACGTC` is not.

In general, a sequence is *form-j*, for $j > 1$, if it is a form-$(j-1)$ or it is a concatenation of a form-$(j-1)$ sequence and a form-1 sequence. For example, `AACCC`, `ACACC`, and `ACACA` are form-3, but `GCACAC` and `ACACACA` are not.

Again, researchers order DNA sequences lexicographically the way we order words in a dictionary. As such, the first form-3 sequence of length 5 is `AAAAA`, and the last is `TTTTT`. As another example, consider the incomplete sequence `ACANNCNNG`. The first seven form-3 sequences that agree with it are:

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

Task

Write a program to find the Rth form-K sequence that agrees with the given incomplete sequence of length M.

Input

The first line contains three integers separated by one space: $M (1 \le M \le 50,000)$, $K (1 \le K \le 10)$, and R $(1 \le R \le 2 \times 10^{12})$. The second line contains a string of length M, which is the incomplete sequence. It is guaranteed that the number of form-K sequences that agrees with the incomplete sequence is not greater than 4×10^{18}, so it can be represented by a `long long` in C and C++ or an `Int64` in Pascal. Moreover, R does not exceed the number of form-K sequences that agree with the given incomplete sequence.

Output

On the first line, print the Rth form-K sequence that agrees with the incomplete sequence in the input.

Sample Input 1

```
9 3 5
ACANNCNNG
```

Sample Output 1

```
ACAAACCCG
```

Sample Input 2

```
5 4 10
ACANN
```

Sample Output 2

```
ACAGC
```

Programming Remark

In C and C++, you should use `long long` data type. The following piece of code show how to read and write a `long long` from and to standard input/output:

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

In Pascal, you should use `Int64`. No special instructions are needed to manipulate data of this type.

Time and Memory Limits

Your program must terminate in 1 second and use no more than 128 MB of memory.

Scoring

The score for each input scenario will be 100% if the correct answer is outputed and 0% otherwise.

In test scenarios worthing 20 points, M will be at most 10.

Roads

The *Kingdom of New Asia* contains N villages connected by M roads. Some roads are made of cobblestones, and others are made of concrete. Keeping roads free-of-charge needs a lot of money, and it seems impossible for the Kingdom to maintain every road. A new road maintaining plan is needed.

The King has decided that the Kingdom will keep as few free roads as possible, but every two distinct villages must be connected by *one and only one* path of free roads. Also, although concrete roads are more suitable for modern traffic, the King thinks walking on cobblestones is interesting. As a result, he decides that *exactly* K cobblestone roads will be kept free.

For example, suppose the villages and the roads in New Asia are as in Figure 1a. If the King wants to keep two cobblestone roads free, then the Kingdom can keep roads (1,2), (2,3), (3,4), and (3,5) free as in Figure 1b. This plan satisfies the King's criteria because (1) every two villages are connected via one and only one path of free roads, (2) there are as few free roads as possible, and (3) there are exactly two cobblestone roads: (2,3) and (3,4).

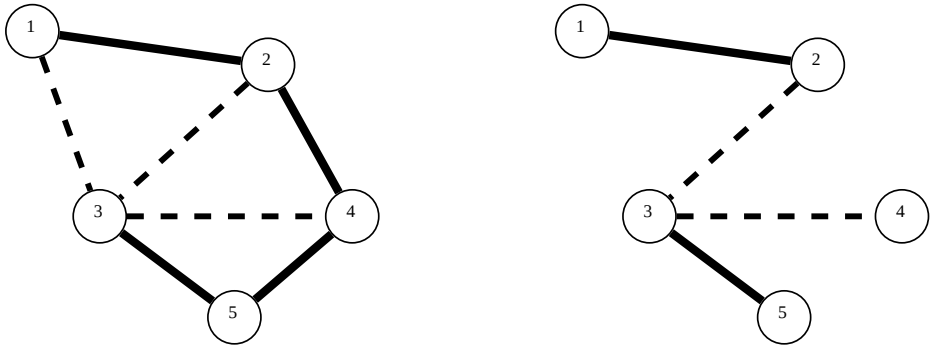

Figure 1: (a) An example configuration of villages and roads in the Kingdom of New Asia. Solid lines denote concrete roads, and dashed lines denote cobblestone roads. (b) A road maintaining plan that keeps two cobblestone roads free. Only free roads are shown.

Task

Given a description of roads in New Asia and the number of cobblestone roads that the King wants to keep free, write a program to determine if there is a road maintaining plan that satisfies the King's criteria, and output a valid plan if there is one.

Input

The first line contains three integers separated by one space:

- N, the number of villages ($1 \leq N \leq 20,000$),
- M, the number of roads ($1 \leq M \leq 100,000$), and
- K, the number of cobblestone roads the King wants to keep free ($0 \leq K \leq N - 1$).

The following M lines describes the roads in New Asia, which are numbered 1 to M. The $(i+1)$st line describes Road i. It contains three intergers separated by one space:

- u_i and v_i, the two villages connected by Road i. Villages are numbered 1 to N, and
- c_i, the type of Road i; $c_i = 0$ if Road i is made of cobblestone, and $c_i = 1$ if it is made of concrete.

There will be no more than one road connecting a pair of villages.

Output

If there is no road maintaining plan that satisfies the King's criteria, your program should print `no solution` on the first line of the output.

Otherwise, your program should output a valid road maintaining plan by listing roads that will be kept free, one road per line. To list a road, print the line in the input that describes it. The roads can be listed in any order. If there are more than one valid plan, you can output any such plan.

Sample Input

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

(This input agrees with Figure 1a.)

Sample Output

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

(This output agrees with Figure 1b.)

Time and Memory Limits

Your program must terminate in 1 second and use no more than 128 MB of memory.

Scoring

The score for each input scenario will be 100% if the correct answer is outputed and 0% otherwise.

In test scenarios worthing 20 points, K will be at most 10.

Solutions

BEADS

We represent each swapper i over conveyor p_i and p_i +1 as a permutation π_i, such that $\pi_i(p_i) = p_i + 1$, $\pi_i(p_i + 1) = p_i$, and $\pi_i(j) = j$ for $j \notin \{p_i, p_i + 1\}$. Thus, for this task, the answer to question $(K; J)$ is

$\pi_J(\pi_{J-1}(\ldots(\pi_1(K))))$

A naive solution that evaluates that permutation for each question runs in time $O(MQ)$, where M is the number of swappers and Q is the number of questions asked.

We can speed that computation up by some preprocessing. For simplicity, we assume that M is some power of 2. Let permutation $\pi_{i,j}$ be the composition of $\pi_j, \pi_{j-1}, \ldots, \pi_i$, i.e.,

$\pi_{i,j} = \pi_j \bullet \pi_{j-1} \bullet \ldots \bullet \pi_i$

For each level $l = 0, 1 \ldots \log M$, we precompute partial composition $\pi_{i2^l,(i+1)2^l}$ for $i = 0, 1, \ldots M/2^l$. There are $M + M/2 + M/4 + \ldots \leq 2M$ such partial compositions. Note that with these partial compositions at hand, one can answer the query by evaluating $O(\log M)$ partial compositions. For example, to compute $\pi_{1,13}$, we need only to compute $\pi_{1,8} \bullet \pi_{9,12} \bullet \pi_{13,13}$.

Another key issue is how to represent each precomputed $\pi_{i,j}$. Note that if we store each partial composition directly as a complete function, we need $\Omega(N)$ space for one of them and this leads to $\Omega(NM)$ memory requirement, which is too much. We can reduce the memory requirement by only store the value $\pi_{i,j}(a)$ for only a such that $\pi_{i,j}(a) \neq a$. We can store them as a sorted list, and it takes $O(\log N)$ time to evaluate one partial composition. As such, Q questions can be answered in time $O(Q \log M \log N)$. Since there are M swappers and at most $\log M$ levels, the memory requirement reduces to $O(M \log M)$. Any solution that meets this time and space bounds receives full score.

DNA

First, we ignore the templates and only count the number of sequences of form-k.

Let $A[i][k][\alpha]$ denote the number of sequences of form-k of length i beginning with α. Note that if we require that the second character β comes before α, possible sequences of form-k are those constructed by concatenating α with form-(k-1) sequences beginning with β. On the other hand, we require that β is α or comes after α, we can concatenating α with any form-k sequence.

Thus, we have the following recurrence

$$A[i][k][\alpha] = \sum_{\beta \geq \alpha} A[i-1][k][\beta] + \sum_{\beta < \alpha} A[i-1][k-1][\beta]$$

This idea can be extended to count the number of sequences that also match the templates. Table A can be computed in time $O(MK)$.

Given A, we can trace back the computation of A to find the required R-th gene sequence.

Roads

We first note one structural property that allows us to use greedy approach to solve this task. Given a graph $G = (V, E)$, where V is the set of villages, and E is the set of roads. We color each edge *e blue* if it is a

cobblestone road and *red* otherwise. This task wants to find a spanning tree of *G* that contains exactly *k* blue edges.

Consider any spanning tree *T*. Note that if we consider subgraph of *T* that contains only blue edges, we get a forest. It is known that for any two forest *F1* and F2, such that *F1* has *k1* edges, *F2* has *k2* edges, and *k2* > *k1*, there exists some edge $e \in F2$ such that $F1 \cup \{e\}$ remains a forest. (See, e.g., Cormen et al., Chapter 16 on matroids.) This implies that to find spanning tree with *k* blue edges, we can never make a mistake by taking in wrong blue edges.

With that fact, there are many ways to find a required spanning tree. We present here one solution.

The algorithm has two rounds. First, we try to find a set of "required" blue edges, i.e., these edges must be in the tree to ensure connectivity. We start by finding all connected components of a subgraph containing only red edges. We then greedily add blue edges joining different components so that the graph is connected. These blue edges added in this round form the starting blue edge set B. Clearly if we need more than *k* edges to connect the graph, we can be sure that there is no spanning tree with the required property.

On the second round, we start with a forest *F* with only edges in *B*. We then greedily add blue edges to *F* while maintaining that the resulting graph *F* remains a forest. We keep adding until we get *k* blue edges in *F*. After that, we complete the algorithm by using red edges to join different connected components in *F*.

If we fail in any steps, it means that no solution exists.

The main data structure that we use in both rounds is the union-find data structure for maintaining disjoint sets.

In the first round, to find connected components of red edges, we just union every pair of components containing end points of red edges. Before adding blue edges *e*, we call two find subroutines to check if *e* joins two different components. If that's the case, we union both components. In the second round we may proceed like the second step of the first round.

To analysis the running time, for each round, we call at most n unions and $O(m)$ finds. Using a set of tree with union-by-rank heuristics ensures that each operation (union or find) runs in time $O(\log n)$. Thus, the algorithm runs in time $O((n + m) \log n)$.

Regulations

August 19, 2006

Asia-Pacific Informatics Olympiad

Following an initial meeting in 2004, delegates from the South Asian / Western Pacific region met during IOI 2006 and drew up concrete plans for a new regional IOI-like competition. The competition is called the **Asia-Pacific Informatics Olympiad (APIO)** and will be held every year, beginning in 2007.

The initial plans for the APIO are outlined below. Delegates will meet again in Croatia at IOI 2007, where they can look back on the first competition and decide what worked, what did not work, and how the contest should be changed for future years.

Regional Delegations

The following sixteen delegations will be invited to participate in the 2007 contest:

> Australia, Bangladesh, China, Chinese Taipei, Hong Kong, India, Indonesia, Japan, Macau, Mongolia, New Zealand, Republic of Korea, Singapore, Sri Lanka, Thailand and Vietnam.

Contest Format

The contest is a one-day Internet contest, with teams competiting from within their home countries. The contest will run for five hours, and will be held on one of the first two Saturdays of May each year.

- Each delegation must run one or more *official contest sites*. All students must sit the contest at an official contest site.
- Each contest site must choose a fixed starting time for the contest. Different delegations may start the contest at different times, and different sites from within the same delegation may also start the contest at different times (e.g., one time for an eastern site and another time for a western site).
- Each contest site must be fully supervised, so that it is clear that students adhere to the contest rules.
- Each delegation may enter any number of students, up to a limit of 100. However, only the top six competitors from each delegation will form the official team (see the *Results* section below).

Students are eligible for this contest if and only if they are eligible for the IOI in the same year.

Contest Host

Although the contest is online, there will be a *host delegation* each year driving the contest to ensure it is both logistically and scientifically successful. The host will rotate from year to year. The responsibilities of the host include:

- Sending invitations and collecting registrations;

- Coordinating task setting and selection (see below);
- Running the contest website;
- Collecting and judging submissions, and coordinating appeals.

The timeline for registrations is as follows. Delegations must indicate to the host *how many students* they are entering at least three months before the competition. They should indicate to the host the *names of the students* at least one month before the competition.

The first two hosts have been decided: Australia will host for 2007, and Thailand will host for 2008. Future hosts will be decided each year at the IOI.

Task Selection

The task setting and selection is done in such a way that everybody has input (and thus everybody shares the burden). The schedule for task setting is as follows; all communications are over e-mail.

- 4 months before: The host issues a call for tasks. This call is to the wider community, not just the region (e.g., the IOI-GA list should be asked). Tasks should be of approximately IOI difficulty, but with more forgiving test data.
- 10 weeks before: Task submissions are due. Each delegation in the region should attempt to submit at least one task (though more is better). Each task should include test data.
- 10–6 weeks before: The host goes through submissions and makes a shortlist of around 10 tasks.
- 6–4 weeks before: The shortlisted tasks are mailed to the leaders from each delegation in the region; together they vote on a final three tasks, plus two backup tasks.
- 4–2 weeks before: Together the leaders edit and refine the official tasks into their final form.
- 2–1 weeks before: Leaders translate tasks into their native languages and submit translations to the host.
- 1 week before: All translations are made available to all leaders; task setting is now complete.

Tasks that were submitted but never used will kept secret, and will be returned to their submitters for use elsewhere.

Results

After judging is complete, the top six competitors from each delegation will form that delegation's *official team*. These official teams will form the official score table, and will be used for deciding upon awards.

Awards will be in the form of certificates, labelled Gold, Silver, Bronze and Participation. The guidelines for awarding Gold, Silver and Bronze will be the same as for IOI. The host will provide the certificates, and will bring them to the IOI that year for distribution to other delegate leaders.

Call for Tasks

ASIA PACIFIC INFORMATICS OLYMPIAD 2008

The Second CALL FOR TASKS

Important dates

Task Submission Deadline	~~**Monday, March 17, 2008**~~	**Tuesday, April 1, 2008**
Shortlist distribution	~~Monday, March 31, 2008~~	Tuesday, April 15, 2008
Announcement on final set	~~Friday, April 11, 2008~~	Tuesday, April 22, 2008
Translated script submission	~~**Friday, April 18, 2008**~~	**Tuesday, April 29, 2008**
Competition day	**Saturday, May 10, 2008**	

The Asia-Pacific Informatics Olympiad (APIO) is an online contest for the South Asian / Western Pacific region. Further information about the APIO, including the contest format and timeline, can be found at http://www.apio2008.or.th/.

The host of the APIO 2008, Thailand Team to the IOI, invites everyone in the APIO community to submit tasks for the APIO 2008 competition, which is held on May 10, 2008. Guidelines for tasks and instructions for submission are given below.

APIO Competition Tasks

APIO tasks are basically the same as tasks in International Olympiad in Informatics, which are typically focused on the design of efficient, correct algorithms. Input and output are to be kept as simple as possible. The submission of novel tasks types not yet seen in any recent programming contests is encouraged. The guidelines of task preparations of IOI should be considered, with no restriction. The documents about submission of IOI2008 tasks, which can be used as reference to create the tasks, can be found at http://ioi.ms.mff.cuni.cz/sc/documents/.

To ensure a fair and interesting competition for all, tasks should satisfy the following conditions:

- The tasks should not have been seen by any potential APIO 2008 contestants.
- The tasks should not have been used in any similar competition.
- The tasks should be solvable by APIO competitors during an APIO contest round.
- The task descriptions should be unambiguous and easy to understand.
- The tasks should be culturally neutral.
- The tasks should be innovative.

What to Submit

A task submission should be contained in one zip file and submitted as described as follows.

- **Statement of the task in English** The description should be written in English. The description should include any intended time and/or memory limits. The document should be submitted in LaTeX, PDF or MS Word format.
- **Description of the desired solution to the task** At least one sample solution should be included, in either C, C++ or Pascal. It is highly desirable to include a short document that
  - outlines how a correct solution works, and
  - discusses other possible solutions that students might think of but should not score full marks (e.g., solutions that are intuitive but incorrect, or solutions that are too slow). This discussion of incorrect solutions is particularly useful when constructing the test data.
- **Suggestions for grading**, test data or ideas for generating test data, the motivation behind the task, and any comments related to the task are also welcome. Test data should consist of a set of input files, and a set of corresponding solution files.

Individual test cases should be weighted equally.

Test data should come with a text file describing how the data was constructed, and/or what each case is designed to test (e.g., "cases #3 and #4 are balanced trees of maximal size, which gives a worst case scenario for the dynamic programming algorithm"; "cases #5 and #6 contain sparse graphs, so that students with poor edge selection routines can still score some points").

- **Extra information**, especially coded solutions and grading suggestions, is highly appreciated.
- **Contact address** (preferably one e-mail address) and background information on the task author(s): affiliation, country, and a description of the author's role in the APIO or national olympiad, including training duties, over the period from APIO 2008 to APIO 2009.

How to Submit

A secure dropbox is also available on the internet at http://www.apio2008.or.th. An authorization code can be obtained from jtf@ku.ac.th or jittat@gmail.com.

Notes

Please note that all material submitted becomes the property of the APIO community, and that by submitting the author is assigning the Copyright and all rights to the submitted materials to the APIO community.

Note that we do not wish to forbid authors of submitted tasks to be involved in other competitions and training, but we do ask them to take all necessary precautions to safeguard confidentiality.

Task selection process

Tasks that are not used will be kept secret, and will be returned to their submitters so that they can be used elsewhere. Note that all submitted tasks will be seen by the host organizing committee (in this case Thailand), and the shortlisted tasks will be seen by all heads of delegations from the region. Communications among heads of delegations and hosts about the APIO tasks should be done mainly by e-mail and secured web.

All submitted tasks will be shortlisted by the host of APIO 2008. Then, the shortlisted tasks are distributed to all heads of delegations. Hosts of delegations should provide suggestions and propose the final set of competition tasks to the host in order to make a final decision on the competition task set. If any countries need to translate the task scripts to their languages, the translated script must be submitted to the host before the day of competition.

At various moments during the preparation of the APIO 2008 competition, it is possible that submitted tasks are dropped from further consideration. However, it may still be necessary to revert such decisions later. In general, all submitted tasks, whether dropped or not, should be kept secret until the end of competition.

Authors of submitted tasks will be recognized during APIO 2008 by listing their name, affiliation, and country (unless they specifically decline this).

Contact person: Punpiti Piamsa-nga (pp@ku.ac.th) and Jittat Fakcharoenphol (jtf@ku.ac.th)

Site Administrator Guide

Site Administrator User Guide

The Asia-Pacific Informatics Olympiad (APIO) 2008

Registration

1. Check whether your site is available at http://sites.apio2008.or.th. If you cannot find your site or you do not have password, you have to contact your country administrator.
2. Make a list of contestant at your site
    a. After you login, you have to just insert your student names. System will generate accounts and passwords for all students automatically.
    b. You have to make a printout of login/password one sheet for one account. These accounts must be distributed to students on the contest day. Please do not inquire accounts before May 8 since you might get passwords for practice session.
3. Prepare the contest room.
    a. A computer must be provided to each contestant.
    b. 32-bit compilers should be installed.
    c. The computer must be connected to the Internet.

On the competition day

1. Inform the students about the contest server. http://evaluator.apio2008.or.th
2. Since students are connected to the Internet, they should be warned about interaction with other people inside and outside the contest site and using inappropriate software during the exam such as instant messenger, e-mail, etc.
3. By regulation of APIO, the exam must be proctored. Proctor must do the same thing as regular exam, including checking the identification. Then, provide the students their accounts and passwords.
4. After all contestants are seated, site administrator has to start the contest manually by logging on site administrator webpage (http://evaluator.apio2008.or.th) and click a start button. After the button is clicked, the contest will run within 5 hours. Intermission and restart is not available.
5. Site administrator and proctors may have to clarify the tasks. Do not explain anything but encourage them to ask a question that you can answer *'yes'* or *'no'*. If it is not about task clarification or you cannot answer yes or no, your choice of answers are:
    - *'answer is in the task script'*
    - *'invalid question'*
    - *'no comment'*

    If your students can write a question in English (or Thai), students can ask us directly via message box on the website; however, we will answer by above policy. Moreover, please encourage students to ask within the first hour of competition.