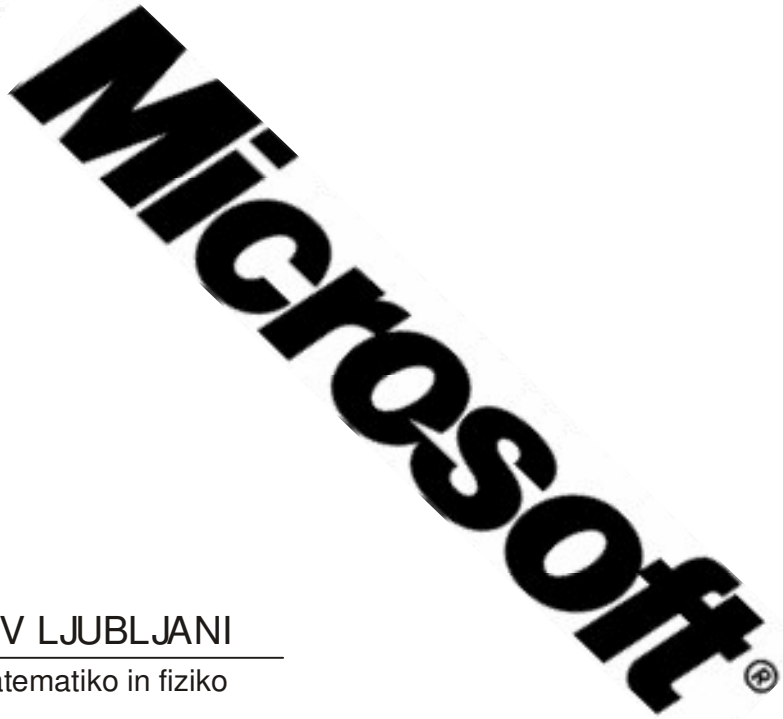
Odprto ekipno prvenstvo UNIVERZE V LJUBLJANI v programiranju

2003

2. krog

Glavni pokrovitelji



UNIVERZA V LJUBLJANI

Fakulteta za matematiko in fiziko

Pokrovitelji

Šolski servis
Janez Vrankar

NAVODILA

Prijava: Prijavite se lahko le na en računalnik. Če bodo med tekmovanjem z njim slučajno težave, javite vodstvu tekmovanja. V primeru, da boste hkrati prijavljeni na dveh računalnikih, bo ekipa diskvalificirana.

Uporabniško ime je sestavljeno iz besede acm in številke računalnika.

Primer: uporabniško ime na računalniku št. 1 je: acm01

uporabniško ime na računalniku št. 12 je: acm12

Geslo je sporočeno pred začetkom tekmovanja.

Vaše domače področje na disku je `/home/acmXX` kjer je XX številka računalnika

Priprava delovnega okolja

Po uspešni prijavi morate odpreti dva terminala in program Netscape.

Terminal št.1: v njem boste pisali programe, prevajali programe in pošiljali programe sodniku.

Terminal št.2: v njem boste dobivali obvestila o pravilnosti rešitev in pošiljali morebitna vprašanja sodniku.

Program Netscape: bo omogočal pregled trenutne uspešnosti ekip

Pisanje programov:

Na voljo so urejevalniki besedil: *vi, kwrite, mcedit, emacs*, *gedit in kedit*

Ime programov je točno določeno pri opisu nalog. Pazite na `return 0,` ki mora zaključiti vsak uspešno izveden C program.

Prevajanje programov:

Za prevajanje programov morate uporabljati naslednje prevajalnike:

```
C      (*.c)        gcc    primer:   gcc -o program program.c
C++    (*.cpp)      g++    primer:   g++ -o program program.cpp
Pascal (*.pas)      gpc    primer:   gpc -o program program.pas
Java   (*.java)     gcj    primer:   gcj -o program --main=program program.pas
```

Pošiljanje programov:

Za pošiljanje programov sodniku uporabljate ukaz `submit`.

Primer: `submit program.c`

Sporočila o napakah – PE in WA

Zaradi različnih razlogov je zelo težko razlikovati med `Presentation Error` in `Wrong Answer`. Ker preverjanje poteka več ali manj avtomatsko, program pogosto ne zna razlikovati med tem, ali ste le pozabili piko v odgovoru, ali pa ste napačno zračunali. Zato tudi `Wrong Answer` včasih lahko pomeni le manjkajoč znak v odgovoru. Praviloma naj bi sporočilo `PE` dobili le v primeru napačno postavljenih presledkov ali ločil in praznih vrstic, ni pa to nujno!

Komunikacija sodniki – ekipa:

V terminalu 2 izvedete ukaz: `ssh svarun.fmf.uni-lj.si`

Za prijavo na računalnik svarun uporabite isto uporabniško ime in geslo, kot ste ga uporabili za prijavo na vaš računalnik. V tem terminalskem oknu se bodo izpisovala sodnikova sporočila vam. Če želite poslati sporočilo sodniku, morate v tem terminalu napisati ukaz:

```
write sodnik
```

in potem sporočilo, ki ga zaključite s kombinacijo tipk CTRL + D na začetku prazne vrstice. Prosimo, da **ne** pošiljate sporočil tipa `... Sedaj smo poslali program ...` in `... Naš program zagotovo dela, pa dobimo WA ...` Ekipa, ki bo pošiljala tovrstna sporočila, je lahko tudi diskvalificirana.

Trenutni rezultati:

Program Netscape: bo omogočal pregled trenutne uspešnosti ekip

Po zagonu programa greste na naslov:

http://svarun.fmf.uni-lj.si/index.html

Kadar želite videti trenutno stanje, morate narediti »reload«.

Pritožbe:

So možne le takoj po zaključku tekmovanja. Kasneje se rezultati lahko spremenijo le v primeru drastične sodniške napake.

Preklapljanje med angleško in slovensko tipkovnico:

Z ukazom v terminalu:

sexkbmap si

setxkbmap us

Nastavitve veljajo za novo odprte programe.

One Person "The Price is Right"

Program price.c, price.cpp, price.java, price.pas

In the game show "The Price is Right", a number of players (typically 4) compete to get on stage by guessing the price of an item. The winner is the person whose guess is the closest one not exceeding the actual price. Because of the popularity of the one-person game show "Who Wants to be a Millionaire", the American Contest Management (ACM) would like to introduce a one-person version of the "The Price is Right". In this version, each contestant is allowed G $(1 \le G \le 30)$ guesses and L $(0 \le L \le 30)$ lifelines. The contestant makes a number of guesses for the actual price. After each guess, the contestant is told whether it is correct, too low, or too high. If the guess is correct, the contestant wins. Otherwise, he uses up a guess. Additionally, if his guess is too high, a lifeline is also lost. The contestant loses when all his guesses are used up or if his guess is too high and he has no lifelines left. All prices are positive integers.

It turns out that for a particular pair of values for G and L, it is possible to obtain a guessing strategy such that if the price is between 1 and N (inclusive) for some N, then the player can guarantee a win. The ACM does not want every contestant to win, so it must ensure that the actual price exceeds N. At the same time, it does not want the game to be too difficult or there will not be enough winners to attract audience. Thus, it wishes to adjust the values of G and L depending on the actual price.

To help them decide the correct values of G and L, the ACM has asked you to solve the following problem. Given G and L, what is the largest value of N such that there is a strategy to win as long as the price is between 1 and N (inclusive)?

Input

The input consists of a number of cases. Each case is specified by one line containing two integers G and L, separated by one space. The end of input is specified by a line in which $G = L = 0$.

Output

For each case, print a line of the form:

```
Case c: N
```

where c is the case number (starting from 1) and N is the number computed.

Sample Input

```
3 0
3 1
10 5
7 7
0 0
```

Sample Output

```
Case 1: 3
Case 2: 6
Case 3: 847
Case 4: 127
```

Head Judge Headache

Program judge.c, judge.cpp, judge.java, judge.pas

You are the Head Judge of the ACM Eastern European Regional Programming Contest. The master Judge computer has been infected by computer virus which has formated the hard disk.

Input

You have only the listing of submitted problems log formatted as follows:

Team_No(≤ 25) *Problem_letter*(`A`...`H`) *Time_of_submittion*(*h:mm*) *Status_of_the_run*(`Y`/`N`)

Output

Write a program to compute the final standing by using the following rules:

- Teams are ranked according to the most problems solved.
- Teams who have solved the same number of problems are ranked by least total time.
- The total time is the sum of the time consumed for each problem solved.
- The time consumed for a solved problem is the time elapsed from the beginning of the contest to the submital of the accepted run plus 20 minutes for each previously rejected run. Further submisions will be ignored.
- There is no time consumed for a problem that is not solved.
- Teams with equal score are assigned equal rank sorted by team number.

Sample Input

```
1 A 0:50 N
3 A 1:12 Y
2 B 1:19 N
1 A 1:20 Y
2 B 1:35 N
1 B 1:36 N
3 B 1:40 Y
4 A 1:40 Y
3 C 1:41 N
```

Sample Output

```
RANK TEAM PROBLEMS TIME
1    3    2        172
2    1    1        100
2    4    1        100
```

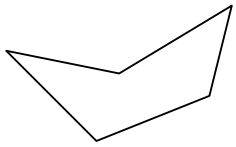
Trgovski center (*KCi naloga*)

Program center.c, center.cpp, center.java, center.pas

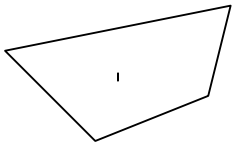
Investitor želi postaviti nov trgovski center. Na voljo ima več potencialnih lokacij. Na stroške postavitve centra močno vpliva cena dovoljenj, ki je odvisna od tega ali je lokacija znotraj mestnih meja ali zunaj.

Mesto je podano kot seznam hiš. Za potrebe naloge predpostavimo, da je vsaka hiša samo točka v ravnini. Koordinate točk naj bodo cela števila. Mesto je vedno konveksno in ker so prebivalci zelo racionalni, so meje mesta ravno konveksna ovojnica vseh hiš. Primer:

Nepravilne meje:



Pravilne meje:



Privzemimo tudi, da je točka, ki leži točno na konveksni ovojnici, znotraj mestnih meja. Mesto ima najmanj eno hišo, prav tako ima investitor na voljo vsaj eno potencialno lokacijo za trgovski center.

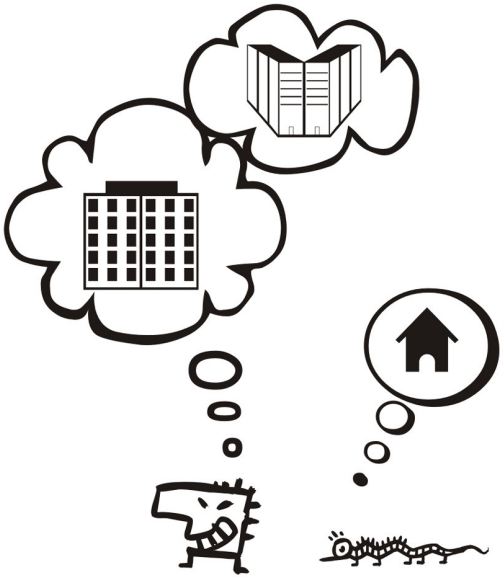
Napišite program, ki bo najprej prebral podatke o hišah, ki pripadajo mestu in nato še podatke o vseh potencialnih lokacijah. Za vsako potencialno lokacijo naj izpiše ali je znotraj mestnih meja ali zunaj. Na vhodu je v prvi vrstici najprej število hiš v mestu, sledijo vrstice s pari koordinat vsake hiše, zadnji hiši mesta sledi število potencialnih lokacij in njemu koordinate vsake potencialne lokacije v svoji vrstici.

Input

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

Output

```
2 2 znotraj
1 0 znotraj
4 1 zunaj
```



Ball Toss

Program ball.c, ball.cpp, ball.java, ball.pas

Classmates stand in a circle facing inward, each with the direction *left* or *right* in mind. One of the students has a ball and begins by tossing it to another student. (It doesn't really matter which one.) When one catches the ball and is thinking *left*, she throws it back across the circle one place to the left (from her perspective) of the person who threw her the ball. Then she switches from thinking *left* to thinking *right*. Similarly, if she is thinking *right*, she throws the ball to the right of the person who threw it to her and then switches from thinking *right* to thinking *left*.

There are two exceptions to this rule: If one catches the ball from the classmate to her immediate left and is also thinking *left*, she passes the ball to the classmate to her immediate right, and then switches to thinking *right*. Similarly, if she gets the ball from the classmate to her immediate right and is thinking *right*, she passes the ball to the classmate to her immediate left, and then switches to thinking *left*. (Note that these rules are given to avoid the problem of tossing the ball to oneself.)

No matter what the initial pattern of left and right thinking is and who first gets tossed the ball, everyone will get tossed the ball eventually! In this problem, you will figure out how long it takes. You'll be given the initial directions of n classmates (numbered clockwise), and the classmate to whom classmate 1 initially tosses the ball. (Classmate 1 will always have the ball initially.)

Input

There will be multiple problem instances. Each problem instance will be of the form

$n\ k\ t_1\ t_2\ t_3 \ldots t_n$

where n ($2 \le n \le 30$) is the number of classmates, numbered 1 through n clockwise around the circle, k (> 1) is the classmate to whom classmate 1 initially tosses the ball, and t_i ($i = 1, 2, \ldots, n$) are each either L or R, indicating the initial direction thought by classmate i. ($n = 0$ indicates end of input.)

Output

For each problem instance, you should generate one line of output of the form:

```
Classmate m got the ball last after t tosses.
```

where m and t are for you to determine. You may assume that t will be no larger than 100,000. Note that classmate number 1 initially has the ball and tosses it to classmate k. Thus, number 1 has not yet been tossed the ball and so does not switch the direction he is thinking.

Sample Input

```
4 2 L L L L
4 3 R L L R
10 4 R R L R L L R R L R
0
```

Sample Output

```
Classmate 3 got the ball last after 4 tosses.
Classmate 2 got the ball last after 4 tosses.
Classmate 9 got the ball last after 69 tosses.
```

Baudot Data Communication Code

Program baudot.c, baudot.cpp, baudot.java, baudot.pas

Data are communicated between digital computers as sequences of bits. To provide meaning to a sequence of bits, the bits are grouped to form a data character and an encoding scheme, or translation table, is provided to allow a computer system to translate each group of bits into a character. The ideal encoding scheme will provide a unique code for every possible character to be communicated and stored in the computer, but this requires that each group have a sufficient number of bits for each data character.

A code used early in the data communications industry is the Baudot code. Baudot uses five bits per character, thus allowing up to 32 distinct characters. As a technique used to extend this limitation, the code uses up-shift and down-shift modes as is used on a typewriter. In the Baudot code, each five bits transmitted must be interpreted according to whether they are up-shifted (figures) or down-shifted (letters). For example, the bit pattern 11111 represents up-shift and the bit pattern 11011 represents down-shift characters. All characters transmitted after the sequence 11111 but before the shifted sequence 11011 are treated as up-shift characters. All characters transmitted after the sequence 11011 are treated as down-shift characters until the pattern 11111 is recognized.

Input

The input consists of two parts. The first part is the Baudot character set: line one contains the 32 down-shift characters and line two contains the 32 up-shift characters. (Note: spaces are inserted for the shift characters.) The remainder of the input file consists of one or more messages encoded using the Baudot code. Each message will be on a line in the input file. Each line will consist of 1's and 0's, with no characters between the bits. There can be up to 80 bits per message. Every message contains at least one character.

The input file will be terminated by end-of-file. The initial state of each message should be assumed to be in the down-shift state.

Output

The output should consist of one line of text for each message. The output should contain the character representation, as translated using BAUDOT, of each of the messages.

Sample Input

```
<T*O HNM=LRGIPCVEZDBSYFXAWJ UQK
>5@9 %,.+)4&80:;3"$?#6!/-2' 71(
10010011001100001001111110111000011111011110 1
0011000011011110010011111000010011000100011001101111000001 11
```

Sample Output

```
DIAL:911
NOV 5, 8AM
```