Odprto ekipno prvenstvo UNIVERZE V LJUBLJANI v programiranju

2003

poskusno tekmovanje

Glavni pokrovitelji

Odprto ekipno prvenstvo
Univerze v Ljubljani
v programiranju
2003

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.

By the Digits of Babylon

Program digits.c, digits.cpp, digits.java, digits.pas

Old Babylonians (2000—1600 BC) are known to be among the first mathematicians. First, it was clear that they used a simple positional notation system with a base of sixty (sexagesimal). They had a sign for 'one' 𒁹 which is just bundled to represent 'two' 𒁹𒁹, 'three' 𒁹𒁹𒁹, and so on, until you get to 'ten', which has a different sign 𒌋. Continue bundling the 'one's and 'ten's until you reach 'sixty', at which point you move over a column and use the 'one' symbol again. In this way, an Old Babylonian mathematician could represent any number using combinations of just two symbols.

Instead of cuneiform symbols, nowadays we usually write ordinary numbers for Babylonian digits and separate them with commas. In this way, number 130 is written as 2,10 which means 2*60+10.

There was one complication, though: ancient civilizations had no concept of zero, and therefore no symbol for it. But you need a zero in a positional system: 120 is written as 2,0. Babylonians sometimes left empty space where 0 should stand, but there is another way out: there is no need for 0 if you admit 60 as a digit. Thus, you write 1,60 instead of 2,0 for number 120.

In this task you will write a program that translates numbers to Babylonian notation.

Input

The input consists of a series of integers (1 <= *n* < 1,000,000,000), one per line. A final 0 marks the end of input.

Output

Write the Babylonian translation of every number (except final 0), one number per line.

Sample Input

```
120
4000
0
```

Sample Output

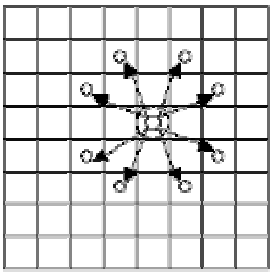
```
1,60
1,6,40
```

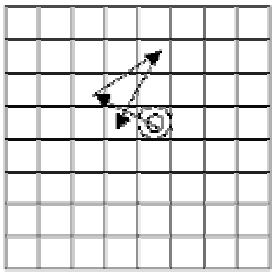
Knight's Charge

Program knight.c, knight.cpp, knight.java, knight.pas

The chessboard is an 8 by 8 grid of squares. For the purpose of this problem consider the rows of the chessboard to be numbered from 1 to 8 starting at the top and moving down, and the columns to be numbered from 1 to 8 starting at the left and moving right. Board positions will be identified by (row, column) ordered pairs.

The knight is a chess piece capable of moving exactly two squares in one direction and then one square perpendicularly. The figure on the left below shows the possible moves for a knight located on the board at (4,5). Assuming the board is empty except for a knight, the problem is to determine the minimum number of moves required to move from point A to point B on the board. The figure on the right below shows one possibility for the minimum required three moves to move a knight from (4,5) to (4,4).





Input

The first line of input contains a single integer that specifies the number of test cases in the remainder of the input. Each test case contains 4 integers in the range 1 to 8, delimited by one or more spaces, representing the row and column of the starting position on the board, and the row and column of the destination position, respectively.

Output

For each test case, simply output the minimum number of moves required to reach the destination position from the starting position.

Sample Input

```
2
5 5 4 5
3 1 3 8
```

Sample Output

```
3
5
```

Word Reversal

Program word.c, word.cpp, word.java, word.pas

For each list of words, print a line with each word reversed without changing the order of the words. Each word consists of English letters only.

Input
The input consists of *T* test cases. The number of test cases (*T*) is given in the first line. Each test case consists of a single line: each line contains a list of words separated by one space.

Output
For each test case, print the output on one line.

Sample Input
```
2
I am happy today
We want to win the first prize
```

Output for the Sample Input
```
I ma yppah yadot
eW tnaw ot niw eht tsrif ezirp
```