Super Computing and Distributed Systems Camp Programming Contest

2010

INSTRUCTIONS

This contest is designed to be solved by teams of at most four (4) students, in a maximum time of four (4) hours. You can use any of C, C++, Java or FORTRAN programming languages, with MPI and/or OpenMP libraries.

Each problem gives you one point if it is correctly solved using MPI, OpenMP or both, ***and*** you provide a sequential version of your solution with the same core algorithm.

The winner team is the team with more problems solved in four (4) hours. If two or more teams have the same number of problems solved, the winner is the team with the best *performance.* Performance is decided by judges, using the speedup formula: $S = \frac{Sequential\ time}{Parallel\ Time}$ running the programs on the same architecture. The best overall Speedup over all the executions is the winner, between the teams with the same number of problems solved.

Feel free to ask the judges about the problems. OpenMP and MPI documentation is available: feel free to ask the judges in any moment.

Have Fun!

PROBLEM A
BOMBING FIELD

Statement

In a very far, far away country, the Military Command (MC) is planning how to destroy an enemy battlefield. However, in the same field is located a populated area, so they must be very careful.

In a simulated scenario, the battlefield is described as a square area of N x N yards, where $100 \leq N \leq 1000$.

In this simulated scenario, Military Targets (MT) are marked as negative numbers, where the number describes the "strength" of the target. So, a -8 target is stronger than an -2 one. Civilian Targets (CT) are marked in the same fashion, but using positive integers. So, a CT with 8 is more important than a CT with 2. Yards with no interest, are marked with 0.

The MC experts program a series of attacks, with special bombs that can destroy square areas. Each of such bombs, have two (2) parameters: the size of the area to destroy and the power itself. *Power* is an integer number, so its effect on the field is to decrease the number of each yard affected by the bomb in "power" units, if the yard is a CT, and to increase the damage, in the case of MT.

After a number of attacks, MC wants to know:

- How many MT where totally destroyed
- How many MT where partially destroyed, i.e., final number is lower than 0
- How many MT were not touched
- Hoh many CT where affected partially by the bombing
- How many CT where totally destroyed
- How many CT where not touched

Your mission is to write a parallel program that helps MC to obtain such information

Input Format

Simulated scenarios values are to be read from standard input. The first number describes the size N of battlefield. In the second line, there is a number T of targets, including both CT and MT. Then, T lines, with three integers: coordinates X, Y ($0 \leq X,Y \leq N-1$) and the value of the target (positive for CT, negative if a MT).

After T lines, a number B describes the number of attacks planned by the MC. Each attack is defined by four numbers: coordinates X, Y ($0 \leq X,Y \leq N-1$), size R of the square radius of the bomb, and the power P of the bomb. Square radius means: given the X and Y coordinates of the bomb, each yard within (X-R,Y-R) and (X+R,Y+R) is affected by the bomb in P units.

The number B of attacks has no limits, and certain X,Y coordinates could be repeated. Bombs have effect only in the battlefield, so coordinates less than 0 or greater than N-1 must not be considered.

It's guaranteed there are at least one CT and one MT. None MT nor CT shares the same X,Y coordinates.

Output Format

After B attacks, you must write on standard output:

```
Military Targets totally destroyed: A
Military Targets partially destroyed: B
Military Targets not affected: C
Civilian Targets totally destroyed: D
Civilian Targets partially destroyed: E
Civilian Targets not affected: F
```

Example

Input:

```
10
4
0 0 8
5 5 100
1 1 -2
7 7 -6
5
2 1 2 3
1 1 1 4
7 7 0 3
6 6 4 8
9 9 8 1
```

Output for the input example:

```
Military Targets totally destroyed: 2
Military Targets partially destroyed: 0
Military Targets not affected: 0
Civilian Targets totally destroyed: 0
Civilian Targets partially destroyed: 2
Civilian Targets not affected: 0
```

PROBLEM B
WEIRD PUZZLE

Statement

A popular cross-word game is to find words inside a square matrix, in any of vertical or horizontal directions. A weird version of this puzzle is when we must find the words even if they are sparse along a line. i.e., find all the letters of the word in the right order in the same vertical or horizontal line, but there could be additional letters in between. For instance, in this version the word `winner` is considered to be in the line "`a r e h n t n i a s w g`"

Moreover, words can be in different directions (right to left, upper to lower and vice-versa) and they can wrap around the matrix, it means that a word beginning at the right most side of the matrix, can finalize at the left side of the matrix, and the same for vertical direction. For instance, the word `contest` is considered to be in the line "`a t e r s t c k o n q`".

Your challenge if to find at most different words as you can, given a square matrix of letters, and a list or words. You must simply have to show the number of different words found.

Input Format

Matrixes and words are to be read from standard input. The first number describes the size N of square matrix of letters, $1 \leq N \leq 1000$. The next N lines have the content of each row of the puzzle, containing N letters of the English alphabet, lowercase.

After the N lines, it is an integer representing the number W of words. Then, W lines, one word per line, lowercase. No entire words will be repeated, and it is guaranteed no words are sub-strings of another one.

Output Format

The output is simply an integer, representing the number of different words you find inside the puzzle. Note that a word could appear more than once, in such case, you have to count *only the first time* the word is found.

Example

Input:

```
5
aocde
adrft
wkdig
wabgt
ooooo
4
ooo
ar
bw
wc
```

Output for the Input example:

```
3
```

PROBLEM C

YET MORE PRIMES

Statement

Primal test is a common task in many applications, specially for security, cryptography, and other interest areas. There are many techniques used to verify if a number is prime, and the common problem is that all techniques consumes lots of time.

The problem is easy to solve, giving the number to be tested and time enough to calculate. But a weird scientist took our numbers, and cut them in two halves, so we have first to re-arrange the numbers, looking for the primes. In this case, we have the certainty that *all* the numbers we have are primes, but the problem is how to re-order them

Input Format

Numbers will be read from standard input. In the first line, there is the number P of prime numbers ($1 \leq P \leq 100$). Then, P lines with the *first half* of the numbers to be tested, with no order. Next, P lines with the *second half* of the numbers, with no order. Prime numbers in this problem are taken as strings, so *first half* and *second half* simply means an arbitrary cut of such string. For instance, prime number 504155039 can be cut in 5041 and 55039, or 504155 and 039

Note that second half numbers could begin with a lead zero, so you must take it into account.

Output Format

You have to print the list of the P prime numbers, in the same order of the *first half*.

Example

Input:

```
4
50415
100
5041
55750
01
0213
5039
55713
```

Output for the Input example:

```
504155039
1000213
504155713
5575001
```