NOI 2010 TASKS OVERVIEW

Tasks
Task 1: CARD
Task 2: BESTP
Task 3: SAILING
Task 4: PKMATCH
Task 5: WEATHER

1. **Each task is worth 20 points.**
2. **Each task, except Task 4, will be tested on 5 sets of input instances. For each set, your program scores 4 points if it produces the correct output for all the input instances in that set.**
3. **The input instances vary in size, ranging from small to sizes around the limits specified in the question. All input instances in the same set have roughly the same size.**
4. **The maximum execution time on each input instance is 15 seconds.**
5. **Sample input and output files for each task, except Task 4, are stored in your home directory. Files with extensions ".in.0" and ".out.1" are the input and output files as given in the task description. Files with extension ".in.1" are additional large input instances. Note that the corresponding output files for ".in.1" are not given.**
6. **Multiple runs of your program on the same input must produce the same output.**

HAPPY PROGRAMMING!

CARD

Alice is a card dealer at the poker table in newly opened ResortWorld casino. As a strange personal quirk, she has two ways of moving a card when she shuffles the deck:

`A`: She takes the card at the top of the deck and moves it to the bottom.

`B`: She takes the second card from the top of the deck and moves it to the bottom.

Initially, Alice has m cards (note that m can be very much more than the standard 52 cards found in a deck) which are labeled consecutively: the card at the top is labeled 0 and the card at the bottom is labeled $m - 1$. Consider a sequence of moves like:

```
ABBABA
```

The table below shows the deck after each move in the sequence is applied.

Next move	A	B	B	A	B	A	
Current deck arrangement	0	1	1	1	4	4	0
	1	2	3	4	5	0	2
	2	3	4	5	0	2	3
	3	4	5	0	2	3	1
	4	5	0	2	3	1	5
	5	0	2	3	1	5	4

In this question we want to know: given a sequence of moves and a k, where $0 < k < m-1$, what is the label of the $(k-1)$–th, k–th and $(k+1)$–th cards from the top of the deck after the entire sequence is applied? Here, we treat top-most card as the 0–th card. In our example above, if $k = 3$, then the answer is "3 1 5".

Input

Your program must read from the standard input. The input consists of m and k, where $0 < k < m - 1$, and $3 \leq m \leq 1,000,000$, and the sequence of moves in a single line. The last character in the input is the period ".", indicating the end of input. The total number of moves is at least 1 and at most 100,000. In our example above, the input is:

```
6 3 ABBABA.
```

Output

Your program must write to the standard output the $(k-1)$–th, k–th and $(k+1)$–th cards from the top of the deck after the entire sequence of card moves have been applied. In our example above, the output will be:

```
3 1 5
```

BESTP

A company rents out various machines each of which has varying performances over time, from unsatisfactory (where it produces large losses by destroying material) to outstanding (when it is very useful and produces large profit). In each time unit, a machine has one of the following seven performances.

"o": Outstanding, makes a profit of SGD 100 in the time unit;
"e": Extraordinary, makes a profit of SGD 10 in the time unit;
"g": Good, makes a profit of SGD 1 in the time unit;
"a": Average, makes no profit and makes no loss;
"b": Bad, makes a loss of SGD 1 in the time unit;
"i": Insufficient, makes a loss of SGD 10 in the time unit;
"u": Unsatisfactory, makes a loss of SGD 100 in the time unit.

The company supplies a potential customer with data about the performance of each machine over time. For example, the data "aabbg" for the first machine and "ggg" for the second machine would say that the first machine can run over 5 time units with two times average, two times bad and one times good performance while the second machine can run over three time units with good performance on each of them. The task is to identify the maximal profit achievable in any contiguous time interval.

In order to help customers with the decision whether to rent one of the machines for some time intervals, write a computer program which reads input as specified below and finds the largest possible profit which can be obtained by renting the machine for a contiguous time interval.

The output corresponding to an input of form "aabbg" should be 1 as only in the last time unit some profit can be made. The output corresponding to "ggbgg" should be 3 as it is profitable to absorb the loss in the middle and to rent the machine over the full runtime, leading to an overall profit of SGD 3. Consider a more complicated example "ggbggbuoeg". The maximum profit is SGD 111, achieved during the interval "oeg". If the machine never perform better or equal to average, a 0 should be returned for the performance.

In general, the task is to make sure that the output is the maximal possible value taken on a interval. In the case of the input "ggue", the possible intervals correspond to "g" (1), "gg" (2), "ggu" (-98), "ggue" (-88), "gu" (-99), "gue" (-89), "u" (-100), "ue" (-90) and "e" (10). Among these, the number 10 is the largest value and should be returned. Recall that the return value is 0 if every interval is evaluated with a negative number.

Note that the input is given for several machines, the data of each of them separated by a comma and a period signaling the end of the data for the last machine. The algorithm must provide for each of these machines the best possible value.

Input

Your program must read from the standard input the following data. The data of various machines are separated by a comma and the data for the last machine is followed by a period; linebreaks and blanks can be ignored. The performance of the machine per time unit is indicated with the letters “u”, “i”, “b”, “a”, “g”, “e”, “o” as explained above.

Example

```
bbaab b,
ggbgg,
ggbgg buoeg,
aaaag ggeeo,
ggggg ggaaa aaggg ggggg
ggggg gaaag gaaag aagaa
ggggg ggggg ggggg ggggg.
```

Output

Your program must write to the standard output a sequence of numbers, one per each line where the first number refers to the best performance of the first machine in a corresponding interval, the second number refers to the second machine and so on. There are as many outputs as there are commas and periods in the input.

Example

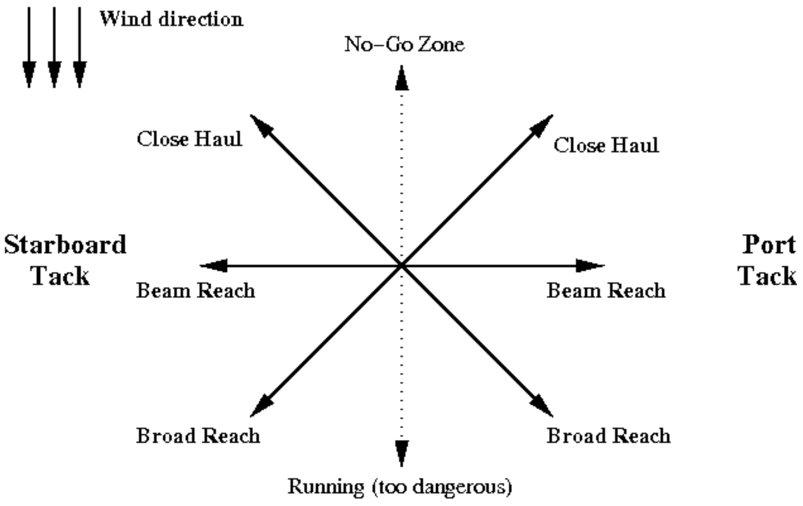
```
0
3
111
123
45
```

SAILING

The Sultan Shoal Regatta, one of the most challenging sailing events in Asia, is held annually in the waters south of Singapore. Anthony, an aspiring young local sailor, named his boat Invincible. He has her trimmed such that she can go in three directions (*points of sail*), relative to the wind, with varying speed:

Allowed point of sail	Explanation
Close haul	Going at an angle of 45^o into the wind
Beam reach	Going at an angle of 90^o perpendicular to the wind
Broad reach	Going at an angle of 135^o compared to the wind

He can do this going either with the wind coming from the left (port tack) or from the right (starboard tack), which leads to six possible directions, as depicted in the following diagram.

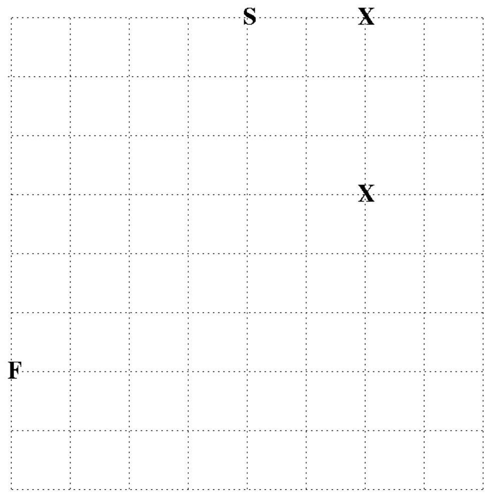


Note that no sailing boat can go directly into the wind for a longer period (No-Go Zone), and that Anthony's sailing skills do not allow him to go directly away from the wind (Running).

Anthony uses his GPS (Global Positioning System) device for navigation and enters waypoints in a grid in shape of a square. With the winds on the day of the regatta, it takes him 10 minutes from a waypoint to the diagonally neighboring waypoint into the wind (going close haul), 9 minutes from a waypoint to the neighboring waypoint perpendicular to the wind (going beam reach), and 15 minutes from a waypoint to the diagonally neighboring waypoint away from the wind (going broad reach).

He instructs the crew to allow a change of direction at each waypoint. At each waypoint, the Invincible can change on the same tack (port tack or starboard tack) from close haul to beam reach, from beam reach to close haul, from beam reach to broad reach and from broad reach to beam reach, all without any loss of speed. Anthony's sailing skills do not allow him to change from broad reach on starboard tack to broad reach on port tack or vice versa (jibing). However, he can change from close haul on port tack to close haul on starboard tack, and vice versa, using a maneuver called *tacking*, in which he loses 12 minutes, due to the loss of speed from pointing the boat temporarily into the No-Go Zone.

The waters south of Singapore have many parked container ships and tankers. Anthony marks those waypoints that he cannot safely reach due to a parked vessel in his GPS device. The following chart contains an example where Anthony has marked the start of the regatta (S), the finish (F), and the unreachable waypoints (indicated with X). At the begining, Anthony is allowed to point the Invincible in any of the allowed points of sail and and reaches the corresponding speed at the start of the regatta. The wind always blows from top to bottom.



Input

Your program must read from the standard input the following data. The first line specifies the length n of the square-shaped grid, where $2 \leq n \leq 150$. Each of the following n lines contains n characters, each of which characterizes the corresponding cell of the grid. The start is marked with with '`S`', the finish with '`F`', the unreachable waypoints with '`X`' and all other waypoints with '`W`'.

Example

The chart in the example above is represented by:

```
9
WWWWSWXWW
WWWWWWWWW
WWWWWWWWW
WWWWWWXWW
WWWWWWWWW
WWWWWWWWW
FWWWWWWWW
WWWWWWWWW
WWWWWWWWW
```

Output

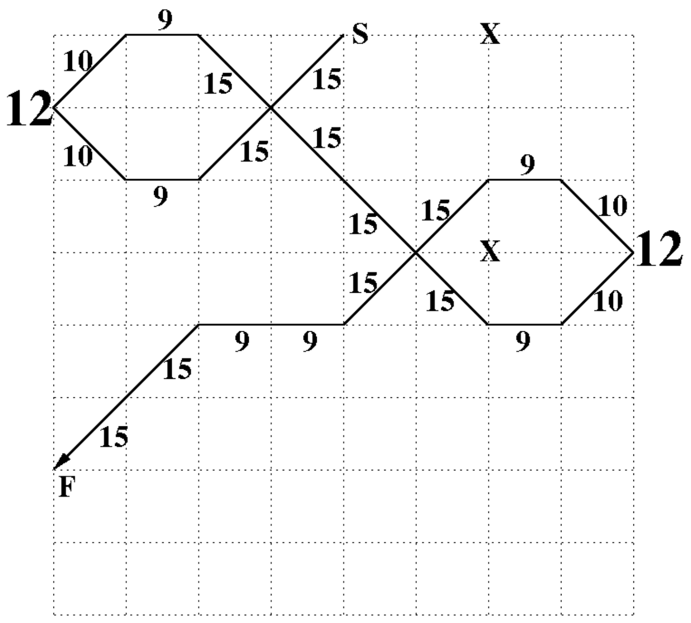
Your program must write one line to the standard output, containing the smallest number of minutes in which Anthony can take the Invincible from the start to sure victory at the finish of the Sultan Shoal Regatta. If there is no way for Anthony to reach the finish, given his sailing skills and the parked vessels, the output line contains the number `-1` instead.

Example

In the above example, the output should be:

```
268
```

One of the routes that allows Anthony to go from start to finish in 268 minutes is indicated below.



In this figure, each section between waypoints is marked with the number of minutes to traverse it. In addition, the two tacks require 12 minutes each, as indicated by the large numbers on the right and left.

PKMATCH

In this two-players card game, each card shows a picture of a monster. There are 6 monsters in the game and they are represented with the integers $\{0, 1, 2, 3, 4, 5\}$.

At the beginning of each game, each player holds 6 cards without duplicates. In other words, each player starts with all 6 monsters on hand. The game is played in rounds. In each round, each player picks a card from his/her remaining cards and shows it to the other player. If both players chose the same card, both cards are removed from the players. Otherwise, the monsters fight and one of them gets knocked-out according to a strength table described below. The knocked-out monster is removed. The game ends when either player has all his cards removed. The player who has cards when the game ends wins the game. If both players have no cards left when the game ends, both players "win", i.e. a "draw" is considered to be a "win".

The strength table indicates which monster get knocked-out when two monsters fight. Suppose monsters i and j fight, we have $S(i, j) = i$ and $S(j, i) = i$ iff monster i gets knocked-out. The following is an example of strength table. Note that $S(i, i) = i$ for all i since both monsters get knocked-out when they are the same. Furthermore, the strength table is symmetric in the sense that $S(i, j) = S(j, i)$ for all i's and j's.

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

Your task is to write a program to play a series of 5,000 games with Randy. A different strength table will be used in each game. Randy always uses a simple strategy: among the remaining cards, it simply randomly picks a card to fight, where each card is equally likely to be chosen. Your goal is to win as many games as possible.

Interactions

In your home directory, you can find two programs: `RANDY` and `interact.sh`. The program `RANDY` is provided by Randy. We now describe the input & output of `RANDY` and how to interact with `RANDY` using `interact.sh`.

Input/Output of `RANDY`

You can "manually" play two games with Randy by issuing the command:

```
./RANDY 2
```

`RANDY` starts the game by displaying the strength table to be used in that game. It first displays 6 lines, each line contains 6 integers. The j-th integer from the left in the i-line is $S(i-1, j-1)$

as shown in the example below. Next, `RANDY` waits for you to enter your choice of monster in the first round, which is an integer. After received your choice, `RANDY` displays its choice in the first round, waits for your choice of the second round, and so on. After the game has ended, `RANDY` displays "`-1`" if you win that game, or "`-2`" if `RANDY` wins that game, and immediate goes to the next game. After all the games have ended, `RANDY` displays the total number of games you have won.

Example

Here is the example of messages exchanged when the command

```
./RANDY 2
```

is issued. For clarity, the input to `RANDY` (in other words, messages sent by you) are shown in boldface.

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

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

Interactions between `RANDY` and your program

The previous example show the input and output of `RNADY`. Recall that your goal is to write a program that interacts and plays a series of games with `RANDY`. Suppose `myprogram` is your executable program, the following command starts the interactions:

```
./interact.sh ./myprogram ./RANDY n
```

where n is the number of games (set to 5,000 during evaluation).

By issuing the above command, messages displayed by `myprogram` will be read by `RANDY`, and messages displayed by `RANDY` will be read by `myprogram`. In other words, the standard output of `myprogram` gets sent as standard input of `RANDY` and vice versa. Since the standard output is "redirected" and not longer sent to the monitor, no message will be displayed on the monitor.

In your home directory, you can find sample C, C++, and Pascal programs, showing how your program can interact with `RANDY`. It is important to note that, in order to interact properly, your program needs to *flush* the standard output after every write operation. See the sample programs for details of flushing. Also note that your program must loop and play the game 5,000 times. Do not submit a program that plays only a single game.

Scoring

`RANDY` will play 5000 games with your submitted program. Your score will be derived from the number of games you win.

number games won	ratio to total number of games	score
≥ 2750	$\geq 55\%$	4
≥ 3150	$\geq 63\%$	8
≥ 3500	$\geq 70\%$	12
≥ 3700	$\geq 74\%$	16
≥ 3800	$\geq 76\%$	20

Note that the expected percentage of wins made by an optimal strategy is more than 77%.

During scoring, the random moves made by `RANDY` and the strength tables are not the same as those generated by the program `RANDY` passed to you. The strength table for the first game is always the one shown in the above example. Subsequent strength tables will be randomly generated.

Programming tips

1. By issuing `interact.sh`, messages sent to the standard output will not be displayed on the monitor. For debugging, you may send messages to the "standard error". Those messages will be displayed. For example,

Pascal:	`Writeln(stderr, 'hello');`
C:	`fprintf (stderr, "hello");`
C++:	`cerr << "hello" << endl;`

However, keep in mind that in the final submitted version, you **must not** write messages to the standard error.

2. It is helpful to note that the program `RANDY` accepts optional second and third parameter.

`./RANDY` n s v

As mentioned before, n is the number of games. The parameter s is the random seed. By changing the random seed, the moves made by `RANDY` will be different. When v is 1, the verbose mode is turned on and `RANDY` will send all messages exchanged and some useful information to standard error. When v is 2, only a single message on the number of games you have won will be sent to standard error. For example, the following uses 105 as random seed and does not turn on the verbose mode,

```
./RANDY 1 105
```

The following command turns on the verbose mode.

```
./interact.sh ./myprogram ./RANDY 1 105 1
```

WEATHER

It is interesting if we can predict the weather for the future N days. Let $A, B, \ldots, Z$ stand for 26 different weather types. Denote the weather types of the future N days be $W[1], \ldots, W[N]$ where $W[i] \in \{A, B, \ldots, Z\}$.

Assume that the meteorologists have a break-through result and they create a machine which can report a list of $N - d + 1$ weather patterns where each weather pattern is the weather types for d consecutive days. For example, assume $N = 10$ and $d = 3$. Suppose the sequence of $N = 10$ days of weather types is $CRSCCCRSRR$. (S, C, and R stand for sunny day, cloudy day, and rainy day, respectively.) Then, the machine will report $N - d + 1 = 8$ weather patterns in alphabetical order. For example, for the above weather sequence, the machine will output: CCC, CCR, CRS, CRS, RSC, RSR, SCC, and SRR. (Note that some weather patterns may occur more than once.)

In this task, for a fixed N and d, given the list of weather patterns generated by the machine, your aim is to compute a weather type for the first day and a weather type for the N-th day, such that both are consistent with the patterns reported by the machine. For the above example, you need to report C, R.

Input

Your program must read from the standard input $N - d + 2$ lines. The first line contains two integers N and d separated by spaces, where $3 \leq N \leq 1000$ and $3 \leq d \leq 20$. Each of the next $N - d + 1$ lines contains weather patterns.

Example

For the above example, the input is:

```
10 3
CCC
CCR
CRS
CRS
RSC
RSR
SCC
SRR
```

Output

Your program must write two characters which represents the weather type of day 1 and day N to the standard output. There is no space in between the characters.

Example

For the above example, the output is:

```
CR
```

Hint

We can formulate the problem as a graph problem. We create a graph G such that every weather pattern $P[1]P[2] \ldots P[d]$ corresponds to an edge between two nodes $P[1] \ldots P[d-1]$ and $P[2] \ldots P[d]$. The problem of reconstructing the weather sequence is equivalent to finding a path covering all edges in G.