Kabul Regional Contest Award Sponsor

icpc global programming tools sponsor

Official Problem Set

2017 ACM/ICPC

The 2017 ACM-ICPC Asia Kabul Regional Contest

A. Sum of Numbers

Razaia and Alireza are school students. They have a homework to add two integer numbers. Your task is to write a program to help them find the result of their operation.

Input:

First line contains an integer **T** which specifies number of test cases. Each test case contains two line each containing an integer.

Output:

For each test case print the addition of two numbers.

Example

Input:

```
1
10
20
```

Output

```
30
```

B. Kabul Post Company

Security issue increases in Kabul cause a few of city's crossroads to be blocked. To ease citizens' movements and avoid problems, Traffic head office publishes a list of open and blocked crossroads every day in the morning. Kabul Posts' postmen use this information to deliver their packages. But it is hard to find a round trip route for delivering the maximum number of packages. Kateb university computer science students developed an application that get the Traffic crossroads report, then it generates a matrix report of entries which represents a crossroad in the city. In the matrix 0-zeros represent blocked crossroads and 1-ones represents open crossroad and *-ones represents package destinations. The application will use this matrix to find out the best possible route that postmen can deliver maximum number of packages in a round trip.

You should rewrite the application with an assumption that postmen always move from top-left to bottom-right and then come back to starting point. Keep in mind that moving to bottom-right, postmen only move to right or down direction. When they come back, they only move left and upward. Whenever a package is delivered, the * sign of crossroad is changed to 1.

Input:

First line contains an Integer **T** which specify the number of test cases. For each test case first line contains two integers **R, C** separated by space which specifies the matrix dimension. In the next lines, there is matrix containing 0s (block crossroad), 1s (open crossroad) and *s (package destination).

Constraints:

$1 \leq T \leq 100$
$1 \leq R \leq 100$
$1 \leq C \leq 100$

Output:

A single integer which indicate the maximum number of packages that can be delivered in a round trip.

Example

Input:

```
1
4 4
****
1001
1001
****
```

Output:

```
8
```

C. Police

You are going to help Kabul police. The Kabul police has many stations in rows to be secured using blocks. Some of these stations have police in them. In order to secure the stations police needs to purchase blocks for their gates (gates have the same size). But Kabul police can make limit number of blocks with varied size. Your task is to write a program to find the minimum number of gates to completely secured the occupied stations. You are given the total number of stations (1<=S<=400), the total number of occupied police stations (1<=P<=S). and, the total number of blocks available to police can make (1<=B<=S).

Input:

First line contains **T** number of test cases. For each test case, the first line contains **S** number of stations. Second line contains **B** number of occupied stations. Third line contains **P** number of available blocks. And the fourth line contains police station numbers that are occupied.

Output:

For each test case print the minimum number of blocked gates.

Example

Input:

```
1
200
3
5
99 100 102 120 130
```

Output:

```
6
```

D. Chess Board

Chess is a mind game and it has millions of players worldwide. It is played by two players, and each player has sixteen pieces. Each piece is whether a king, queen, rook, bishop, knight or a pawn. Each piece move differently than others. And they can capture each other. Check is a situation which a piece can capture opponent king in the next move but the opponent can avoid that by whether moving his/her king or block it with another piece. Checkmate is a situation which the player can't get out his/her king from check position.

Chess is played in world tournaments. Many players from around the globe gather and compete with each other. The Chess federation, records every game so that they can analyze or share the game play with other interested parties. To do so, they use a notation called Portable Game Notation (PGN). PGN contains different part including game title, players, date, address, and …. PGN has a set of special symbols, characters and rules to keep track of each piece movements (movetext).

1. f4 d5 2. Nf3 c5 3. e3 Nc6 4. Bb5 Bd7
5. O-O e6 6. b3 Qc7 7. Bb2 f6 8. c4 Nce7
9. Nc3 Nh6 10. Rc1 Bxb5 11. Nxb5 Qd7
12. Qe2 Nc6 13. cxd5 exd5 14. e4 O-O-O
15. e5 a6 16. Nc3 b5 17. a4 b4 18. Nd1
Kb7 19. exf6 gxf6 20. Bxf6 Re8 21. Ne3
Rg8 22. Qd3 Ng4 23. Nxg4 Qxg4 24. Rf2
Qd7 25. Ne5 Nxe5 26. Bxe5 Rc8 27. Qf3
Kb6 28. d3 Bh6 29. R2c2 d4 30. a5+ Kb5
31. Bc7 "White's last move is one of the most beautiful ever played on the chess-board." Reinfeld

Figure 1 - PGN Movetext Example

The federation wants to have a computer software to analyze each game by giving the game's PGN as input. But for now, they need a program that take the PGN movetext only as input and give them the board (including the position of each piece) as output. We want to develop that software but we don't need it to be a graphical program. We just want it to take PGN movetext (formatted in SAN) as input and give us a 2D matrix that contains pieces (each coded with a unique symbol) at their current location on the board.

Note:

- Bottom left side of the chess board is a1 (as shown in below picture)
- White start the game
- All moves are valid so you don't need to check their validity
- Test cases only include ordinary chess moves (including castlings) except promotion and, en passant.

- Translations
    - King, شـاه
    - Queen, وزیر/ملکه
    - Rook, رخ/قلعه
    - Bishop, فیل/اسـقف
    - Knight اسـب/شـوالیه
    - Pawn سـرباز/پیاده
- *Check out the attachment on how the chess is played.

Movetext Rules:

The movetext describes the actual moves of the game. This includes move number indicators (numbers followed by either one or three periods; one if the next move is White's move, three if the next move is Black's move) and movetext Standard Algebraic Notation (SAN).

For most moves the SAN consists of the letter abbreviation for the piece, an x if there is a capture, and the two-character algebraic name of the final square the piece moved to. The letter abbreviations are K (king), Q (queen), R (rook), B (bishop), and N (knight). The pawn is given an empty abbreviation in SAN movetext, but in other contexts the abbreviation P is used. The algebraic name of any square is as per usual algebraic chess notation; from white's perspective, the leftmost square closest to white is a1, the rightmost square closest to the white is h1, and the rightmost (from white's perspective) square closest to black side is h8.

In a few cases a more detailed representation is needed to resolve ambiguity; if so, the piece's file letter, numerical rank, or the exact square is inserted after the moving piece's name (in that order of preference). Thus, Nge2 specifies that the knight originally on the g-file moves to e2.

SAN kingside castling is indicated by the sequence O-O; queenside castling is indicated by the sequence O-O-O (note that these are capital Os, not zeroes, contrary to the FIDE standard for notation). Pawn promotions are notated by appending = to the destination square, followed by the piece the pawn is promoted to. For example: e8=Q. If the move is a checking move, + is also appended; if the move is a checkmating move, # is appended instead. For example: e8=Q#.

Pieces' Codes:

`k|K` –> white|BLACK King
`q|Q` –> white|BLACK Queen
`r|R` –> white|BLACK Rook
`b|B` –> white|BLACK Bishop
`n|N` –> white|BLACK Knight
`p|P` –> white|BLACK Pawn
`_` –> Empty Cells (underscore)

Input:

First line contains **T**, an integer number which indicate the number of test cases. Each next line contains a SAN string.

Output:

Print the board and the pieces according to given SAN string.

Constraints:

- Execution time: 2 secs

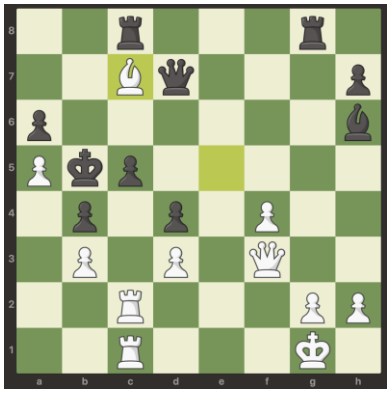
Example

Input:

1

1. f4 d5 2. Nf3 c5 3. e3 Nc6 4. Bb5 Bd7 5. O-O e6 6. b3 Qc7 7. Bb2 f6 8. c4 Nce7 9. Nc3 Nh6 10. Rc1 Bxb5 11. Nxb5 Qd7 12. Qe2 Nc6 13. cxd5 exd5 14. e4 O-O-O 15. e5 a6 16. Nc3 b5 17. a4 b4 18. Nd1 Kb7 19. exf6 gxf6 20. Bxf6 Re8 21. Ne3 Rg8 22. Qd3 Ng4 23. Nxg4 Qxg4 24. Rf2 Qd7 25. Ne5 Nxe5 26. Bxe5 Rc8 27. Qf3 Kb6 28. d3 Bh6 29. R2c2 d4 30. a5+ Kb5 31. Bc7

Output:



```
__R___R_
__bQ___P
P_______B
pKP_____
_P_P_p__
_p_p_q__
__r___pp
__r___k_
```

E. Grouping Symbols

Kabul National Security Office recently arrested a terrorist that has a secure message for other terrorists in their group. This messaged was secured by a combination of "<>؟|/+_)(&^%!@×$#*" characters. It has been known that every one of these symbols represent an alphabet and a combination of them will make a word. The National Security Agents could discover some combinations of these characters that make a word. Based on their discovery, they want to convert an encoded string into groups of symbols that make a word.

Your job is to write a program which takes the groups of symbols that make up words, an obscure string, then print all possible groups of symbols that make a word after disjoint from the encoded string.

Input:

First line contains an integer **T** which indicate the number of test cases. Each test case contains two line. First line contains groups of characters that each represent a word. And the second line contains the encoded messages.

Output:

Print all the possible ways that the encoded message can be cracked using the given character groups.

Example

Input:

```
1
*&% #!$% * {)}|! *&%&(%){ &(%){ )}/ {)} |!
*#!$%*&%&(%){)}/{)}|!
```

Output:

```
* #!$% *&% &(%){ )}/ {)}|!
* #!$% *&%&(%){ )}/ {)}|!
* #!$% *&% &(%){ )}/ {)} |!
* #!$% *&%&(%){ )}/ {)} |!
```

F. Personal Currency Exchange

There are a few micro finance services in Kabul. They provide different investment and business services for their customers. Currency exchange is one of those services. You start with a base amount in a specific currency, and then you can exchange (buy or sell) it with other currencies.

Ahamd just started using one of these services, he had a few thousand Afghani as his initial capital. He had exchanged it with other currencies over last two weeks. Now, after two weeks Ahmad has a few thousand Afghani as profit. He wants to know how to earn even more profit. So, he decided to find the profit for each exchange transaction and find out which day he could have the most profit.

Write him a program that take Ahmad's exchange transactions as input and give him the max amount of profit and the day that he could have that.

Input:

First line contains T, an integer number which indicate the number of test cases. For each test case, the first line contains number of transaction (N). The second line contains Ahmad's initial capital in a specific currency (base currency) separated by "|" sign from exchange rates of other currencies (1 base currency is equal to ? other currency). And N next lines contains currency exchange rate changes and transactions separated by "|" sign in chronological order (first line is for the first day, second line is for second day and so on).

Note:

- Base currency is the currency which other exchange rates given based on that.
- Buy and sell rate is the same
- Initial capital doesn't have decimal point
- Currency list is limited to AFN, USD, EUR, IRR, PUK
- Transaction format: [amount][source currency]=[target currency], eg: 123AFN=USD
  It means you have exchanged 123AFN with x USD.
- Transaction amounts are integer numbers, and currency exchange rates and changes have at most 6 digits in decimal point.
- In each day first currency exchanges occurs, then transactions take place.
- Currency rates may not change in some days
- "NO_TRANSACTION" is printed for the days that no currency rate change and transaction is happened.

Output:

Print the amount of profit in base currency (2 digits max in decimal point) separated by space from day number which Ahmad could have the maximum amount of profit in the given period.

Constraints:

- Execution time: 2 secs

Example

Input:

```
1
4
58746AFN|EUR=0.013262,USD=0.014534,PUK=1.522070,IRR=59.523809
NO_TRANSACTION
PUK=0.167647,IRR=-8.010997
EUR=0.002131,USD=-0.000751,PUK=-0.020950,IRR=4.962031|1431AFN=PUK,1866AFN=USD,2USD=IRR,0USD=EUR
EUR=0.002027,USD=-0.000925|235PUK=EUR,705IRR=EUR
```

Output:

```
29.26AFN 4
```

G. Petrol Station

Eimal is working in a Petrol station. His duty is to calculate the total Petrol liters has been sold after N customers (1<= N <= 40). The price of each liter of petrol can be changed for each customer. Help Eimal with writing a program to add up the amount of soled petrol and its related price.
The price can be ranged from 1 to 100.

Input:

First line contains number of test cases **T**. Each test case contains three lines. First line contains **N** Total number of customers. Second line contains amount of petrol purchased by each customer **N1, N2**, …. And the third line contains corresponding price **N1, N2**, ….

Output:

For test case print the **Total petrol sold**, **Total Price**.

Example

Input:

```
1
3
20 25 40
10Af 20Af 30Af
```

Output:

```
85, 1900
```

H. Maximum Number

Hamid has three bags containing balls printed with different numbers (1<=N<=9). Every bag has balls with different colors say (Green, Red, Blue). Hamid wants to find the maximum number from all the possible combinations formed from these balls starting from last to first input.
The maximum number Hamid is looking for has following properties. In case no combination is found return NONE on output.

1. Divided by any even numbers between (1 to 10) including first and last.
2. Divided by Three

Input:

First line contains **T** number of test cases. For each test case, first line contains **N** bag size. Second line contains **N1** first bag tagged number. Third line contains **N2** second bag tagged number and fourth line contains **N3** third bag tagged number.

Output:

Maximum number

Example

Input:

```
1
2
1 2
1 2
1 2
```

Output:

```
222
```

I. Book Reading Competition

In a book reading competition, there are N books that should be read. For answering questions for every book, it is necessary to read the whole book. A high number of books have been introduced and it is not possible to read all of them. So, every contestant should select and read a few number of books according to the time he or she has.

Your job is to write a program to find maximum number of questions that can be answered by the contestant using following information:

- Amount of time that contestant can put on reading the book
- Average reading speed ratio (number of page per hour) of every contestant
- List of books and their page counts
- Maximum possible speed (number of page per hour) for reading every book
- Number of question that is selected from each book

Input:

The first line of input contains an integer **T** denoting the number of test cases. Then **T** test cases follow. Each test case consists of six lines. The first line consists of **N** the number of books. The second line consists of **D**, the maximum time which contestant can spend for reading. In the next line an integer or decimal **P**, denoting the average speed of contestant in reading based on pages per hours. The next line contains **W** space separated positive integers denoting the number of each book's questions in contest and in the fifth line are **V** space separated positive integers denoting the number of each book's pages. Finally, the sixth line are **Y** space separated positive integers denoting possible speed for reading each book.

Constraints:

$1 \leq N \leq 100$
$1 \leq D \leq 100$
$1 \leq W[i] \leq 100$
$1 \leq V[i] \leq 100$

Output:

Print the **maximum possible** questions that can be solved within given conditions that you can obtain for each test case in a new line.

Example

Input:

```
2
3
40
2
1 2 3
40 50 10
2 2 2
4
50
4
5 8 3 5
100 120 90 100
4 3 2 5
```

Output:

```
5
10
```