A+ Computer Science
Computer Science Competition
Hands-On Programming Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Point Values and Names of Problems

Number	Name
Problem 1	Rings
Problem 2	Vows
Problem 3	RSVPs
Problem 4	Venue
Problem 5	Straws
Problem 6	Vendors
Problem 7	Playlist
Problem 8	Honeymoon
Problem 9	Seating Chart
Problem 10	Servers
Problem 11	Making the rounds
Problem 12	The Big Day

For more Computer Science practice tests and materials, go to www.apluscompsci.com

1. Rings

Program Name: Rings.java **Input File: rings.dat**

You have started a new business "All In Wedding Planning". You are a wedding planner who takes care of literally everything, and you just landed your first clients. You have picked the rings for the couple, and you need to send them a picture, but your camera on your phone is broken. You need to print out an ASCII mock up so they will get the basic idea.

Input

Read in the ring ASCII art as shown below OR simply print the ASCII art picture. The ASCII ring art data file is exactly as shown below. If you do not know how to read in a data file, you can just type up the code to print the ASCII art ring.

Output

Output the ring ASCII art as shown below.

Assumption

The student output and the judge's output for this problem are EXACTLY the same.

Example Input File

```
   ,-==-,
   \/\/\/
    \\//
   .-''-.
 .'.-""-.'.---.
/ /    .'\ \--.'.
| |   / /| |   \ \
\ \  | | / /    | |
 '.'-:\ \.'    / /
   '---; '-..-'.'
        `'----'
```

Example Output to Screen

```
   ,-==-,
   \/\/\/
    \\//
   .-''-.
 .'.-""-.'.---.
/ /    .'\ \--.'.
| |   / /| |   \ \
\ \  | | / /    | |
 '.'-:\ \.'    / /
   '---; '-..-'.'
        `'----'
```

2. Vows

Program Name: Vows.java **Input File: vows.data**

You now need to help come up with wedding vows for your first clients, (or more specifically the groom). You need to print out the vows as shown below.

Input

There will be one data set in the date file. The data set will consist of one set of vows followed by a single string. The vows will contain an unknown number of lines. The lines may have whitespace. You must read in all lines in the vows. Immediately following the lines of the vows, there will be a single string. There will be NO spaces in the string. The string may contain uppercase letters, lowercase letters, numbers, and punctuation marks. You will print the vows with every occurrence of `BLANK` in the vows replaced with the string that follows the vows.

Output

Output the vows you read in with `BLANK` replaced with the string that follows the vows.

Example Input File

```
I, BLANK, take you, BLANK, for my
lawful BLANK, to have and to hold
from this day forward, for better,
for worse, for richer, for poorer, in
sickness and in health, until death
do us part. I will love and honor
you all the days of my life
SALLY
```

Example Output to Screen

```
I, SALLY, take you, SALLY, for my
lawful SALLY, to have and to hold
from this day forward, for better,
for worse, for richer, for poorer, in
sickness and in health, until death
do us part. I will love and honor
you all the days of my life
```

3. RSVPs

Program Name: RSVPS.java **Input File: rsvps.dat**

You need to write a program to count up how many people have RSVP'd yes and no to your first client's wedding. You will be given a number of letters to catalogue, and need to determine the total number of yes's and no's, and the percentage of those invited who will actually be attending your wedding.

Input

The first line will contain a single integer `n` that indicates the number of letters to catalogue. Each letter will consist of the party's name (a string containing no spaces), the size of the party (an integer), and a Yes/No denoting whether or not that party will be attending the wedding.

Output

Output a line containing two integers, denoting the number of no's and the number of yes's, followed by a percentage (rounded to two decimal places), denoting the percentage of people who will be attending your wedding out of those invited, all separated by spaces.

Example Input File

```
5
Johnson 4 Yes
Whitaker 2 No
Coyle 5 Yes
Hatley 3 Yes
Wade 1 No
```

Example Output to Screen

```
12 3 80.00%
```

4. Venue

Program Name: Venue.java **Input File: venue.dat**

The groom for your first contract is very hung up on the wedding venue, and how much each square foot of the place will cost. You need to print out the cost per square foot per hour for each of the venues given so that the groom can go over it.

Input

The first line will contain two single integers `m` and `h`, indicating the number of data sets that follow, and the number of hours you are looking to rent the venue for. Each data set will consist of a string (the venue name, containing no spaces), a double (the cost for the venue), and two integers (the dimensions of the venue, length and width respectively), all separated by spaces.

Output

For each venue, output the name, followed by a space, followed by the price per square foot per hour rounded to 6 decimal places.

Example Input File

```
3 8
RDR 10000.00 400 800
Cadillac_Ranch 19450.00 1000 1000
Party_City 1810.00 1200 850
```

Example Output to Screen

```
RDR 0.003906
Cadillac_Ranch 0.002431
Party_City 0.000222
```

5. Straws

Program Name: Straws.java **Input File: straws.dat**

At the wedding, it is decided that drawing straws will be used to decide who cleans up. Before the straws can be drawn, one big straw has to be cut into optimal pieces. You decide you need to go visit your guy The Big Sammy Hammy at his new small business so he can help cut up the big stick into smaller pieces. Sammy has opened a cutting business where he is given big items and then cuts them in such a way to maximize the value of each smaller piece. He will use his cutting system to help you out. For his system, each different size of cut obtains a certain value, so determine the maximum value that can be obtained from the given initial big straw's length if it is cut optimally. This will help you determine who is cleaning up once the straws are drawn.

Input

The first line will contain two integers. The first value `n` states the number of initial big straws to chop up. The second value `r` that states how many straw values will follow. The following line will contain `r` integers denoting the values for each of the lengths of straws, with the index in the list denoting the length of straw corresponding to the given value. For example: in the sample input, the integer `1` in the array corresponds to a straw of length `1`, the integer `5` corresponds to a straw of length `2`, and so forth. The following `n` lines will each contain one integer denoting the length of the initial big straw to be cut.

Output

Output a single integer denoting the maximum value that can be obtained from cutting the given big straw.

Example Input File

```
4 8
1 5 8 9 10 17 17 20
4
6
7
8
```

Example Output to Screen

```
10
17
18
22
```

6. Vendors

Program Name: Vendors.java **Input File: vendors.dat**

Your next task for your clients is to get some vendors that are available and fit within the budget given by the clients. You will be given a list of possible vendors for a specific thing (DJ, Photography, Catering, Cake, etc.), along with the price for each, their average review rating (0.0-5.0), and a list of dates (MM/DD) where they are unavailable. You need to remove those who are unavailable on the day of your clients' event, and those who are over budget, and sort the rest based on average review rating (highest to lowest), then price (lowest to highest), then alphabetically by vendor name.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will begin with 1 line containing a string `s`, an integer `m`, a date in the format `MM/DD`, and a double `p`, denoting the item you are looking for, the number of vendors for this item, the date of the wedding, and the maximum price budgeted for this item, respectively, all separated by spaces. Each of the following lines will contain a string denoting the name of the vendor, followed by 2 floating point numbers denoting the price of the vendor and their average review rating, respectively, followed by an unspecified number of space-separated dates `MM/DD` denoting dates where that vendor is unavailable. There are not guaranteed to be any days the vendor is unavailable.

Output

For each data set, output the name of the item first on its own line, followed by a colon. This should be followed by up to 5 lines, denoting the ordered list of qualified vendors for this item in the following format: `(place): (name) (price) (average review rating)` Price should be rounded to 2 decimal places, and average review rating should be rounded to 1 decimal place. Place should be 1-5, if there are less than 5 qualified vendors, then there should be less than 5 printed. If there are no viable vendors, output the starter line, followed by the string "`No Viable Options.`" on its own line.

Example Input File

```
2
Catering 4 03/26 5000.00
Valley_Forge_Catering 4500.00 4.5 03/26 03/27
Planters_Diner 5200.00 4.5 04/03
Olive_Garden 4800.00 4.3 03/27
Neighborhood_Restaurant 4200.00 4.2 04/04 05/05
DJ 3 03/26 3500.00
Eds_DJ_Service 3200.00 4.9 03/27 03/29
TPain 100000.00 5.0 03/30 03/31
Valley_Forge_DJ 3500.00 4.8 03/25 03/27
```

Example Output to Screen

```
Catering:
1: Olive_Garden 4800.00 4.3
2: Neighborhood_Restaurant 4200.00 4.2
DJ:
1: Eds_DJ_Service 3200.00 4.9
2: Valley_Forge_DJ 3500.00 4.8
```

7. Playlist

Program Name: Playlist.java **Input File: playlist.dat**

You now need to create a playlist for the wedding. You have found a list of common wedding songs online, and you need to update this list using the clients' requests for do not plays and must plays. You need to write a program to determine the list of songs to be played at the wedding. You will make rounds of adjustments to the playlist, and send it to the clients, who will then add new rules. You have also been given a maximum number of songs, which is how many songs are in your original playlist when you first sent it to the clients. Songs should be added to the top of the playlist in the order they are given, and removed from the bottom when necessary. Having less than the maximum number of songs is not a problem. All song names will be unique, and all strings will be case-sensitive.

Input

The first line will contain a two integers `n` and `m`, denoting the number of songs in the initial playlist (and maximum number of songs in the playlist), and the number of rounds of adjustments to be made. The next n lines will each contain a song entry for the original playlist, in the following format:
`(song name): (artist) (genre)`, where all 3 values are strings containing no spaces, denoting the name of a song, artist who sings it, and genre of the song. The `m` rounds of adjustments to be made will follow. Each round will begin with an integer, `c`, denoting the specific changes the clients would like you to make to the list before sending it back to them. Each of the following `c` lines will contain a change in one of the following formats:

- `REMOVEN [name]` – remove this song from the current playlist.
- `REMOVEG [genre]` – remove all songs of this genre from the current playlist.
- `REMOVEA [artist]` – remove all songs by this artist from the current playlist.
- `BANN [name]` – remove this song, and ban it from the playlist.
- `BANG [genre]` – remove all songs of given genre, ban all future songs of this genre.
- `BANA [artist]` – remove all songs by given artist, ban all future songs by this artist.
- `ADD [name] [artist] [genre]` – Add given song to the top of the playlist, if this increases the size of the playlist too much, remove the song at the bottom.
- `ROTATE N` – rotate the playlist `N` places top to bottom.

Output

Output the names of all the songs in the playlist, in order from the top of the playlist to the bottom, after each round of changes, separated by commas.

Example Input File

```
6 2
DNA: Kendrick_Lamar Rap
Everlong: Foo_Fighters Alternative
Runaway: Bon_Jovi Rock
TNT: AC/DC Rocks
Dedicate: Lil_Wayne Rap
Numb: Linkin_Park Alternative
2
ROTATE 2
REMOVEN Runaway
2
BANG Alternative
ADD Motiv8 J.Cole Rap
```

Example Output to Screen

```
Dedicate, Numb, DNA, Everlong, TNT
Motiv8, Dedicate, DNA, TNT
```

8. Honeymoon

Program Name: Honeymoon.java **Input File: honeymoon.dat**

The clients have decided that they want to go on a honeymoon to Europe and travel by train. You need to determine the minimum cost for train passes for the dates they need to travel. You can buy a 1, 5, 10, or 30-day train pass. These passes cover n days of consecutive travel on the train. You will be given all the days in the year the clients need to travel, numbered 1-365.

Input

The first line will contain a single integer `n` that indicates the number of data sets to follow. Each data set will consist of two lines of space separated integers. The first line will contain 4 integers, denoting the price of a 1-day, 5-day, 10-day, and 30-day pass. The second line will contain an unspecified number of integers denoting the days (1-365) of the calendar year in which a train pass will be necessary. The line of days is guaranteed to be in increasing order, and all values will be unique.

Output

For each test case, output the minimum cost to buy train passes to cover all the necessary days.

Example Input File

```
3
2 6 9 15
1 4 6 7 8 20
2 6 9 15
1 2 3 4 5 6 7 8 9 10 30 31
2 6 9 15
1 2 3 4 5 6 7 8 9 10 11 12 13 14 30 31
```

Example Output to Screen

```
10
13
17
```

9. Seating Chart

Program Name: Seating.java **Input File: seating.dat**

You now need to help the bride with the seating chart (she only wants to help a little bit with this). You will be given a list of people to seat at a table, as well as a matrix outlining how each person feels about each other person (1 for friends, 0 for ambivalence, -1 for dislike/hatred). For each seating arrangement you will calculate a "score" by adding the value in the matrix corresponding to each pair of people sitting next to each other. You need to determine, for each group of people at a table, how many different combinations there are that maximize the "score" of the group, and what all these groups are. Keep in mind, we are counting unique combinations (ABC, BCA, CAB are all the same combo, but for our purposes ABC and ACB are NOT the same. Basically if you can rotate one permutation into another they are the same, but reversals are not the same).

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will begin with an integer `m` (`4 <= m <= 10`) denoting how many people are in this group. The following `m` lines will each contain `m` space-separated integers denoting the matrix, `con`, containing the values explaining how a particular person feels about each other person at their table. `con[i][j]` denotes the integer explaining how person `i` feels about person `j`.

Output

For each test case, output the maximum attainable "score" for each table, followed by a space, followed by how many possible combinations there are that obtain that "score".

Example Input File

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

Example Output to Screen

```
2 2
4 2
```

10. Servers

Program Name: Servers.java **Input File: servers.dat**

The big day is approaching! You are rushing around town getting things ready, but your current issue involves the servers route from the kitchen to the catering station in the venue. You need to determine the minimum number of people the server can encounter on their path, as every person they need to pass will take time and the food will get cold if they take too long. You have a map laid out, of the venue, divided into a grid, with an estimated amount of people in each square based on your extensive knowledge of weddings. Keep in mind the servers can only move down and right.

Input

The first line will contain a single integer `n` that indicates the number of test cases to follow. Each test case will begin with a line containing two integers, `r` and `c`, denoting the number of rows and columns in the grid. The next `r` lines will each contain `c` integers denoting the number of estimated people in that square of the grid.

Output

For each test case, output the minimum number of people a server can encounter on a path from the top left of the grid to the bottom right.

Example Input File

```
2
3 3
1 3 1
1 5 1
4 2 1
2 3
1 2 3
4 5 6
```

Example Output to Screen

```
7
12
```

11. Making the Rounds

Program Name: Rounds.java **Input File: rounds.dat**

You are working on your last few tasks before the wedding, and your next one is a doozie. You need to determine the shortest amount of time for the bride and groom to "make the rounds" during dinner to go see everyone they need to see, given that they have to speak to everyone at a particular table every time they stop to talk to even a single person at that table, and if you pass through a table space you must stop to talk. You also need to keep in mind that they will travel slower through certain sections of the venue, such as the dance floor, photo booth, and bar. There will also be certain stations they need to go to, such as cake cutting or outdoor photos, that they will need to go to as well. Only visiting one of the locations for each type of required station is necessary. The lists of two people for each table are distinct from each other. You have been given a layout of the venue upon which to plan out these routes, which has been split into a grid, and will be made up of the following characters:

- S – denotes the point in which the couple is sitting (the starting and ending point for their journey), also takes 10 seconds to move through.
- . – denotes an empty space in the venue, which will take 10 seconds to travel through.
- 0-9 – denotes the location of a table in the venue (there will never be more than 10). Each table will take the standard 10 seconds, plus the time it takes to talk to everyone there at the table.
- A-Z – denotes a specific location in the venue (could be a photo booth, bar, dance floor). These spaces will each take 40 seconds to travel through, unless they are required, then they will take 2 minutes.
- # - denotes an impassable object in the venue, such as a flower wall or DJ stand.
- $ - denotes a particularly chatty guest who is not sitting at a table (never required to speak to them, but it will take 45 seconds to move through their grid section).

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with 2 space separated integers, `r` and `c` (`1 <= r, c <=30`), denoting the number of rows and columns in the map of the venue, and `e` (`0 <= e <= 9`), denoting how many different tables there are. The next line will contain a space-separated list of uppercase characters, denoting the locations in the venue that the couple must make it to before completing their rounds. This line will be blank if there are no required uppercase characters/places to vist. The following `e` lines will each contain a list of integers in the following format: `rnum1,rnum2,…,rnumk|nnum1,nnum2,…,nnumj`
`rnumk` denotes the amount of time needed to speak to the `k`th person at this table you are required to speak to, and `nnumj` denotes amount of time (in seconds) needed to speak to the `j`th person at this table. The tables will be listed in numerical order (the first table listed is shown as 0 in the map, and so on). The following `r` lines will each contain `c` characters denoting the layout of the venue. Note: there will never be more than 8 required stops (tables + locations).

Output

For each data set, output the length of time it takes to travel the shortest path through the venue for the couple when "making the rounds", while speaking to everyone they are required to speak to, and visiting all the locations in the venue that they need to. This value should be listed in minutes, rounded to 2 decimal places.

Example Input File

```
2
6 6 6
D
30|45,50,55
|60,60,60,60
|100,120,30,40
100,65,70|35
|120,120,70,80
85,95|130,65
#S..BB
......
.2.3.D
4.5.DD
.6.1.D
#...JJ
4 3 3
D J
|100,30,40
65,75|10
100|120,35
S.J
.1.
2.3
.D.
```

Example Output to Screen

```
18.25
12.08
```

12. The Big Day

Program Name: Day.java **Input File: day.dat**

The day is here! You have put together quite the event with lots of things going on and stuff to do at the wedding (you kind of did it huge it's a bit like a carnival). Some of the guests want your help determining a schedule to have the most fun doing things at the wedding. You have a start and end time for each event at the wedding, as well as the "value" of this event in the eyes of the person whose schedule you are creating. You want to maximize the total "value", but someone cannot attend more than one event at a time. One event starting at the same time another ends is not considered overlapping.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will begin with an integer `m`, (`0 < m <= 1000`) denoting the number of events at the wedding. The following 3 lines will each contain `m` space-separated integers each denoting the start times, end times, and values of each event, respectively. Event `i` will be described by `start_time[i]`, `end_time[i]`, and `value[i]`. Start times, values, and end times will be in the lower bound and upper bound inclusive range [`0, 10^6`]. Start time will always be less than or equal to end time for an event (some events are very quick like a photo booth).

Output

Output the maximum total value that can be obtained from the given list of events.

Example Input File

```
2
4
1 3 6 2
2 5 19 100
50 20 100 200
4
1 2 4 5
3 5 6 7
60 50 70 30
```

Example Output to Screen

```
250
130
```