Beer Run

General information

The challenge exercises are intended for students who are done with the regular weekly homework exercises and seek an extra challenge. Only those who passed the challenge program entry test will be considered participants of the challenge program.

Your submission for all the parts of the exercise should be in Java. All the challenge exercises must be done individually. Feel free to send an e-mail to course2ip90+CP@gmail.com if you have any questions.

Assignment description

One of your hobbies (next to programming of course) is the annual beer run. Each year you go on an annual beer run which is a race where participants race through a field. There are beers hidden in this field. Each beer has a score attached to it. The point of the race is to run through the field and drink beers, the participant that has the highest score at the end wins. Using your programming prowess you write couple programs that give you the an optimal route through the field so you can win the race.

The input to your programs start with which part it is (1,2, or 3), followed by the size of the grid and then the part specific input.

Part 1

You managed to find a beer-o-meter on the internet which tells you where the beers are hidden and which score they have. Your beer-o-meter shows that the beers are hidden in the grass in a grid pattern and which score each grid has. Additionally, there are directional constraints which you need to abide when navigating through the field. You start in any grid in the most left column. You are only allowed to move east, north-east, or south-east. The first part of your program calculates the max beer score you can get, using legal moves only.

Example

input:

```
Part 1
3 3
1 3 4
7 2 1
4 1 8
```

output:

```
17
```

The optimal route here was 7, 2, 8.

Part 2

Using the same beer-o-meter you join the second event of the beer run. During the second event there is an announcer which on the fly announces which direction you are allowed to go after you drink a beer. But the beer-o-meter recently had a firmware update which allows the beer-o-meter to read the announcer's mind and knows which directions the announcer will call out. If the announcer calls east, you are allowed to go to a cell east, north-east, or south-east of your current cell. The same goes for south, north, and west, mutatis mutandis. You are also allowed to visit the same cell multiple times, this results in the same yield. The amount of directions is equal to the amount of beers you are allowed to drink.

The input to the second part of the program starts with Part 2, then the size of the grid, the grid itself, and how many directions the announcer will call out followed by the directions themselves.

Example

input:

```
Part 2
5 5
8 4 5 1 2
6 1 2 4 7
9 5 4 7 6
2 1 0 7 8
9 5 8 7 4
6 E E S E N E
```

output:

```
53
```

An optimal route starts in cell 3,5 and follows the following path:
E,NE,SW,NE,NW,SE through values 8,7,8,7,8,7,8.

Part 3

======

The last event of the beer run is done on a unicycle. The unicycle allows you to turn in any direction you want and you are allowed to go any direction you want. There are a couple restrictions however, you only get the score of a cell the first time you visit it. If you end up on the same cell in the grid multiple times you only get the points once. Just like part 2 there is a maximum amount of turns you can make, and you can start in any cell you want.
The input to the third part of the program starts with Part 3, then the size of the grid, the grid itself, and how many turns you are allowed to make.

Example

input:

```
Part 3
4 4
9 2 7 4
2 8 3 7
5 1 2 4
```

```
	1 9 8 3
	3
```

output:

```
	31
```

An optimal route starts in cell 0,0 and follows the following path:
SE,NE,SE through values 9,8,7,7.

Handing in the assignment

========================
Hand in a single file named BeerRun.java. Keep efficiency in mind when designing your solution.