Mr. Bellavia

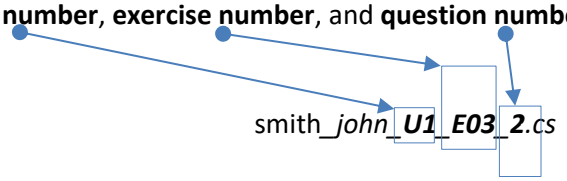
ASSIGNMENT 01 – 2D ARRAY PROBLEM SOLVING

IMPORTANT: Before submission, make a copy of your **'Program.cs'** file for each question and then rename each file to the following:

File Names:

- *last_name_first name*_U4_A01_1.cs
- *last_name_first name*_U4_A01_2.cs

Note: Along with last name and first name, make sure the end of the filename (i.e., before the **.cs**) has the **unit number**, **exercise number**, and **question number.** For example:



Problem 1: Fertile Land

Problem Description:

Consider piece of land that is sectioned into a grid:

7	2	1	6	3
10	5	3	4	10
0	2	10	9	2
10	8	7	7	5
4	7	2	1	5

Each integer in the grid above represents a scale of how fertile the land is for farming (1 being the lowest, and 10 being the highest).

You are to create a small farm in the shape of a cross consisting of 5 squares that will yield the most fertile land. Using the above grid, we would have the following:

Mr. Bellavia

7	2	1	6	3
10	5	3	4	10
0	2	10	9	2
10	8	7	7	5
4	7	2	1	5

Maximum Fertility Yield = 8 + 10 + 2 + 7 + 7 = 34

Input Specification:

The first two lines are two integers M (M > 0 and M <= 1500) and N (N > 0 and N <= 1500) representing the number of grid columns and rows respectively. The following N lines are the rows of the grid each containing M integer values separated by a space, where each integer is between 1 and 10 representing the fertility yield of that piece of land.

Output Specification:

The output will be the maximum fertility yield possible based on a cross shaped farm placed on the grid as described in the problem description.

Sample Input 1:

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

Sample Output 1:

```
34
```

Explanation:

The position (3, 1) on the grid above is the center of a cross placed on the grid that yields the maximum possible fertility yield = 8 + 10 + 2 + 7 + 7 = 34

Mr. Bellavia

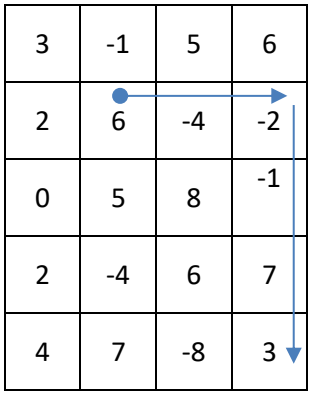
(OPTIONAL) Problem 2: Coin Collector

Problem Description:

Consider a grid of coins and collectors. Each cell in the grid can hold up to 10 coins. A person traveling the grid will collect the coins as they land on each cell. Instead of coins, some cells contain a 'collector' who can remove coins from a person should that person land on that cell. It is also possible for a person to possess a negative number of coins if they land on a collector cell who takes more than they have (for example, if a person currently has 4 coins but the collector takes 5, that person will then have -1 coins).

When travelling the grid, a path chosen by a person can start on any cell, but they must end at the bottom rightmost cell of the grid. There is only one movement allowed: from the chosen starting cell the person first travels all the way right until they cannot go right anymore, and then all the way down until they are at the bottom rightmost cell on the grid. Each cell can contain a positive number representing a number of coins, a negative number representing a collector, or a '0' representing no coins. The numbers in the cells traversed by the person on their chosen path are added together for a total value.

Here is an example of a possible path taken by a person starting at row 2, column 2:



Total coins = 6 + (-4) + (-2) - 1 + 7 + 3
= 9

Write a program that calculates the largest possible number coins that can be gathered by any valid path.

Input Specification:

The first two lines are two integers M (M > 0 and M <= 1500) and N (N > 0 and N <= 1500) representing the number of grid columns and rows respectively. The following N lines are the rows of the grid each containing M integer values separated by a space, where each integer is either positive (coin), negative (collector), or zero (empty).

Output Specification:

An integer representing the largest possible number of coins that can be collected by any valid path.

Sample Input:

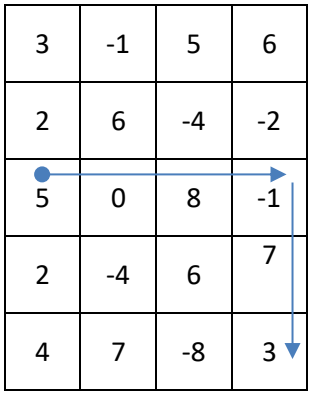
```
4
5
3 -1 5 6
2 6 -4 -2
5 0 8 -1
2 -4 6 7
4 7 -8 3
```

Sample Output:

```
22
```

Explanation:

The path that yields the largest number of coins is shown below:



Total coins = 5 + 0 + 8 + (-1) + 7 + 3

= 22