National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Design and Analysis of Algorithms	Course Code:	CS2009
	Degree Program:	BSCS, BSSE	Semester:	SPRING 2023
	Exam Date:	Tuesday, April 11, 2023	Total Marks:	11 + 19 = 30
	Section:	ALL	Page(s):	3
	Exam Type:	Mid-Term - II		

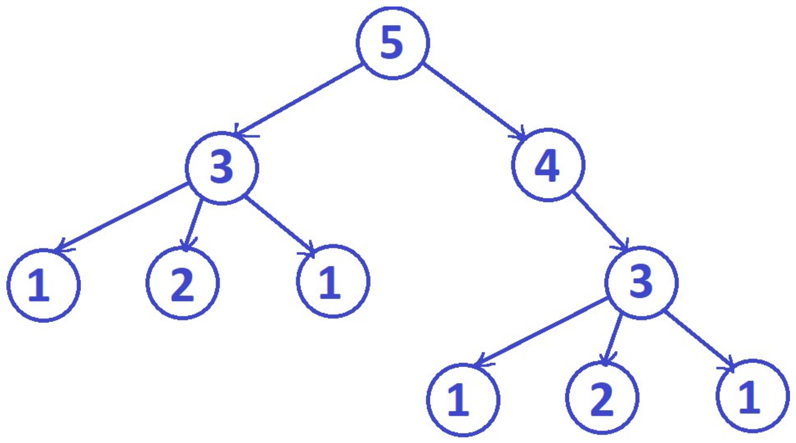
Student : Name:________________________________ **Roll No.**_________________ **Section:**________

Instruction/Notes: Attempt all questions. There are two questions, don't forget to check the back side as well.

Question 1: [3 + 8 = 11 Marks]

a) Draw **recursion tree** of **MyAlgo** using n = 5.
Only the recursion tree is required. No calculations are needed. **There will be no partial credit for this part.**

```
MyAlgo(n)
{
    IF(n == 2)
        return 3;
    ELSE IF (n == 1)
        return 1;
    ELSE IF (n%3 == 0)          // n is a multiple of 3
        return  MyAlgo($\lfloor \frac{n}{2} \rfloor$) - MyAlgo(n-1) + 2 x MyAlgo(n-2);
    ELSE IF (n%2 == 0)          // n is even
        return MyAlgo(n-1);
    ELSE
        return MyAlgo( (n+1) / 2 ) - MyAlgo(n-1);
}
```



b) Convert the pseudocode given above (in part **a**) into bottom-up dynamic programming (iterative code). For this part only **pseudocode** is required.

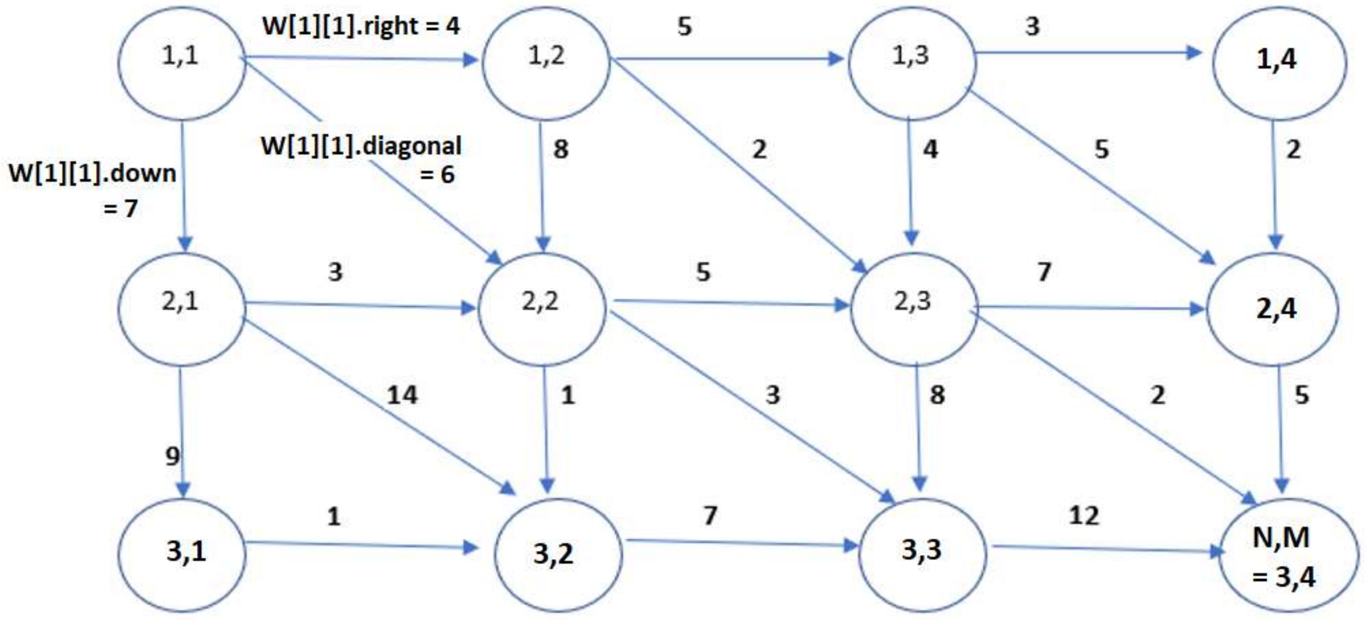
One marks for array of correct size
One marks for base cases
One marks for the loop (3 to n)
One marks for look-up from the array
One marks for storing each value in the array
One mark each for all the cases inside the loop
Total 8 marks

```
A[1..n]
A[2] = 3
A[1] = 1
FOR (i = 3 to n)
        IF (i%3 == 0)
                A[i] = A[ i/2 ] - A[i-1] + 2 x A[i-2]
        ELSE IF (i%2 == 0)
                A[i] = A[i-1]
        ELSE
                A[i] = A[ (i+1)/2] - A[i-1]
RETURN A[n]
```

Question 2:

Assume that there is a moving object of pacman in a weighted grid of (N rows and M columns). Pacman starts from the cell (1,1) and can move a maximum distance of 1 unit cell at a time either in rightwards, downwards or in the diagonal cell, as shown in the figure below. Note that the object can only move forward and there is no way back. At each move pacman can score some points. These points for each move is stored in the array W[1…N][1…M].

If pacman is at cell (1, 1), it can move towards rightwards to cell (1, 2) and earn 4 points which is stored in W[1][1].**right**, or it can move downwards to cell (2, 1) and earn 7 points which is stored in W[1][1].**down** or it can move diagonally to cell (2, 2) and earn 6 points which is stored in W[1][1].**diagonal**. Pacman wants to earn maximum points while going from cell (1,1) to cell (N,M).

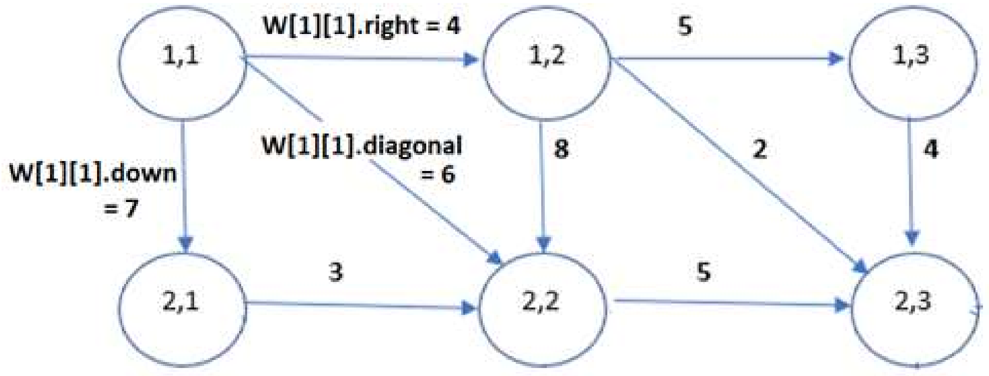


a) **COUNTER-EXAMPLE** [CLO-3] [7 marks]

Consider the greedy algorithm where pacman selects the maximum value among right, down and diagonal from each cell and move in that direction until it reach the cell (N, M), for example at cell (1, 1), pacman will move down because W[1][1].**down** is greater than W[1][1].**right** and W[1][1].**diagonal**.

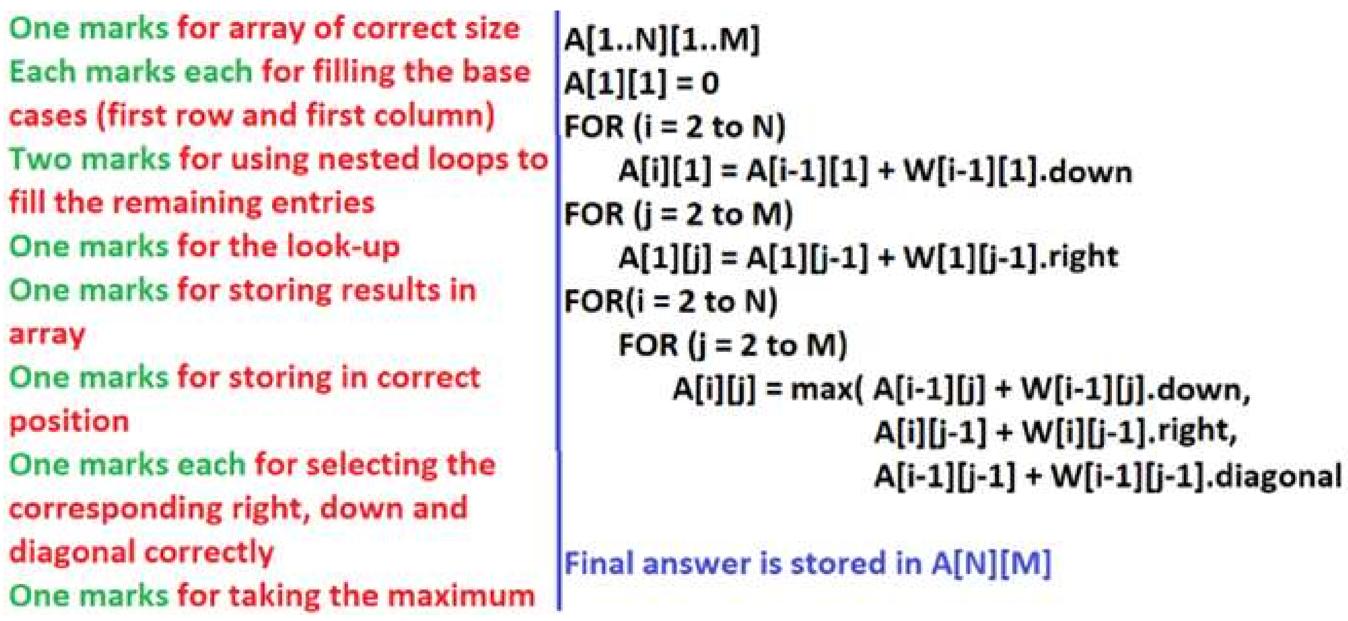
Prove that the greedy algorithm is not correct by providing a counter example.

Only a counter example is required. **There will be no partial credit for this part**. The size of your counter-example should not be larger than 3 x 3.



b) Dynamic Programming [CLO-1] [12 marks]

Design and bottom-up dynamic programming algorithm (iterative algorithm) that calculates the maximum score pacman can achieve starting from the source cell (1,1) to destination cell (N,M). Weights of edges are given in a 2-D array W[1..N][1…M]. Each entry of this array holds three values i.e. W[i][j].right, W[i][j].down and W[i][j].diagonal. For this part only **pseudocode** is required.



One marks for array of correct size
Each marks each for filling the base cases (first row and first column)
Two marks for using nested loops to fill the remaining entries
One marks for the look-up
One marks for storing results in array
One marks for storing in correct position
One marks each for selecting the corresponding right, down and diagonal correctly
One marks for taking the maximum

```
A[1..N][1..M]
A[1][1] = 0
FOR (i = 2 to N)
    A[i][1] = A[i-1][1] + W[i-1][1].down
FOR (j = 2 to M)
    A[1][j] = A[1][j-1] + W[1][j-1].right
FOR(i = 2 to N)
    FOR (j = 2 to M)
        A[i][j] = max( A[i-1][j] + W[i-1][j].down,
                       A[i][j-1] + W[i][j-1].right,
                       A[i-1][j-1] + W[i-1][j-1].diagonal

Final answer is stored in A[N][M]
```