Lab Final

Program: ***B.Sc. in CSE***
Department of Computer Science and Engineering
Southeast University
Semester: ***Spring-2022*** Course Title: ***Advanced Algorithm Lab*** Course Code: ***CSE2032.1***
Total Marks: ***50*** Date and Time: ***06/21/2022*** Slot: ***11:00 AM – 01:30 PM***
Time: ***2.5 Hours (150 Minutes) + Extra 0.5 Hours (30 minutes) for Submission***

Instruction: *Submit this assignment in .PDF format. Take pictures of your assignment in any suitable formats (.JPG, .PNG etc.) and convert all of them in one .PDF file and make sure the picture orientation is straight. The full assignment should be Handwritten.*
ALSO ALL THE CODES MUST BE HANDWRITTEN. SCREENSHOT OF CODE SNIPPET IS NOT ALLOWED.

Submission Deadline: ***02:00 PM, Tuesday (June 21, 2022)***
Submission Pole: *Submit it as Assignment on the same Question Pole on Google Classroom.*

DON'T WASTE YOUR VAUABLE TIME FOR OTHERS! YOU WON'T GET IT BACK ONCE IT'S GONE!

1. Convert the following Pseudo-code to actual coding in any of your preferred programming Language (C/C++/Java will be preferable from my side!)

Declare variables named as i, j, r, c, VAL
Print "Enter the value of r: "
Input a positive integer from the terminal and set it as the value of r
Print "Enter the value of c: "
Input a positive integer from the terminal and set it as the value of c
Declare a 2D matrix named as CM using 2D array such that its dimension will be r x c
Input an integer number (>0) for each cell of CM from terminal and store it into the 2D array
Print the whole 2D matrix CM
Set VAL to CM[0][0]
Set both i and j to 0
While i doesn't get equal to r minus 1 OR j doesn't get equal to c minus 1
    Print "(i, j) → " // i means the value of i and j means the value of j
    If i is less than r minus 1 and j is less than c minus 1
        If CM[i] [j + 1] is less than or equal to CM[i + 1][j], then increment j by 1 only
        Else increment i by 1 only
    Else if i equals to r minus 1, then increment j by 1 only
    Else increment i by 1 only
    Print "(i, j)" // i means the value of i and j means the value of j
    Increment VAL by CM[i][j]
    Print a newline
Print the last updated value of VAL

6 + 2 + 2 = 10 Marks

The above Pseudo-code gives solution to of one of the well-known problems we have discussed in this course. Can you guess which problem it is? Also, can you say to which approach the above Pseudo-code does indicate? Is it Dynamic Programming or Greedy? Justify your answer with proper short explanation.

2. Consider the following 5 x 6 area where each cell is indicated in the form [i, j]:

[0, 0]	[0, 1]	[0, 2]	[0, 3]	[0, 4]	[0, 5]
[1, 0]	[1, 1]	[1, 2]	[1, 3]	[1, 4]	[1, 5]
[2, 0]	[2, 1]	[2, 2]	[2, 3]	[2, 4]	[2, 5]
[3, 0]	[3, 1]	[3, 2]	[3, 3]	[3, 4]	[3, 5]
[4, 0]	[4, 1]	[4, 2]	[4, 3]	[4, 4]	[4, 5]

The journey **starts from the *BOTTOM LEFT CELL and ends at the TOP RIGHT CELL***. The following moves are valid for any REACHING cell [i, j]: (REACHING cell [i, j] means Destination Cell/The Cell ***To*** where you are going!)

- If the i value of any REACHING cell [i, j] is ODD, then you can COME TO/REACH that cell FROM the cells situated at STRAIGHT to its LEFT and STRAIGHT to its DOWNWARDS

- If the i value of any REACHING cell [i, j] is EVEN, then you can COME TO/REACH that cell FROM the cells situated at STRAIGHT to its LEFT, STRAIGHT to its DOWNWARDS and DIAGONALLY DOWNWARDS to its LEFT

Now if I Calculate how many ways are there to reach the ***TOP RIGHT CELL*** from the ***BOTTOM LEFT CELL*** using the ***Dynamic Programming approach***, I am getting an output just as shown below: 15 Marks

1	7	26	70	155	301
1	5	14	30	55	91
1	4	9	16	25	36
1	2	3	4	5	6
1	1	1	1	1	1

Now, write down a code in any of your preferred programming Language (C/C++/Java) that implements the above scenario with the proper deployment of the given moving constraints as the logic of your code.

****** Hints: Take input the number of rows and number of columns of the area from the terminal. You can use for loop or while loop to implement the given moving constraints as your logic. Your code must have a statement that Prints the total number ways to reach the TOP RIGHT CELL from the BOTTOM LEFT CELL. ******

3. Consider the following pattern searching/matching scenario:

Suppose, your text is: DAABAABABADBAABA and the pattern is: AABA. The matching process should be like this:

1st character of the text resides at text[0], 2nd character resides at text[1], 3rd character resides at text[2] and so on. Same goes for the given pattern also, that is, 1st character of the pattern resides at pat[0], 2nd character of the pattern resides at pat[1], 3rd character of the pattern resides at pat[2] and so on. The whole text is divided into some fixed length windows and this window length is same as the length of the given pattern.

Each time the given pattern is compared with a window and it is determined whether the pattern matches with the current window or not. If yes, then print the position and slide forward to the next window (to the right side in the text by one position) and repeat the same process. If not, then just slide forward to the next window (to the right side in the text by one position) and repeat the same process.

This type of matching is referred to as the Naïve Pattern Searching. We may the depict this type of searching in the following way:

	D A A B A A B A B A D B A A B A	
1st window: (DAAB with AABA)	*A A B A*	*Pattern doesn't match in position 0*
	D A A B A A B A B A D B A A B A	
2nd window: (AABA with AABA)	*A A B A*	*Pattern found at position 1*
	D A A B A A B A B A D B A A B A	
3rd window: (ABAA with AABA)	*A A B A*	*Pattern doesn't match in position 2*
	D A A B A A B A B A D B A A B A	
4th window: (BAAB with AABA)	*A A B A*	*Pattern doesn't match in position 3*
	D A A B A A B A B A D B A A B A	
5th window: (AABA with AABA)	*A A B A*	*Pattern found at position 4*
	D A A B A A B A B A D B A A B A	
6th window: (ABAB with AABA)	*A A B A*	*Pattern doesn't match in position 5*

And so on……

5 Marks

Construct a program in any programming language for the given scenario of pattern matching (C/C++/Java is preferable).

****** Hints: Take input the text and the pattern from the terminal. Calculate the length of both the text and the pattern. Let them be t and n respectively. So, there will be t – n + 1 number of search iterations/windows. ******

4. Figure out the Burrows-Wheeler Transform (BWT) for the following text:

"***mynicknameisABC***"

You have to follow the instructions given below before starting to apply the Burrows-Wheeler Transform (BWT):

Replace the ***ABC*** part of the text given above with your SURNAME or NICKNAME**.** For example, if someone's surname/nickname is ***JAMEY***, then above text will become "***mynicknameisjamey***" as you have to put all the characters of your SURNAME / NICKNAME in lowercase letters.

Now, perform Inverting Burrows-Wheeler Transform to reconstruct the original text back using the Last-to-First (LF) Mapping mechanism.

5 + 5 = 10 Marks

5. Construct a Red-Black Tree for the following list of elements and calculate the Black-Height value for each node of the tree:

10 Marks

33 17 50 76 60 82 91 110 95 55 20 5 49