National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Programming Fundamentals	Course Code:	CS1002
	Program:	BS (CS, DS, SE)	Semester	Fall 2022
	Duration:	180 Minutes	Total Marks:	10+10+10+10+30
	Paper Date:	20 December 2022	Page(s):	12
	Section	All Sections		
	Exam Type:	Final Exam		

Student Name: Section:

Registration #:

Instruction/Notes: Answer in the space provided. Show all working. Good luck! **NO ROUGH SHEETS**

Q1. **Write the output of the following:** CLO-2

Output:

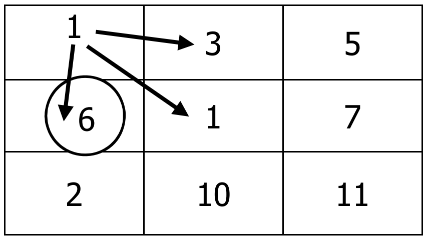
```
void getIndices(int x, int &b)
{
    x = b/2;
    b = x + 1;
}
void replace(int &p, int &q)
{
    p = p + x;
    q = p - q;
    x = p + q % 3;
}
void print(int arr[], int size)
{
    for(int i=0; i<size; i++)
    {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int main()
{
    const int size = 8;
    int x = 0, y = size-1;
    int arr[] = {2,5,3,7,9,4,6,8};
    for(int i=0; i<size/2; i++)
    {
        getIndices(x,y);
        replace(arr[y], arr[x]);
        x++;
        y*=2;
    }
    print(arr,size);
}
```

Q2. Given a `M x N` matrix filled with non-negative numbers, write a function CLO-3

```
int traverseMatrix(int m[M][N], int M, int N)
```

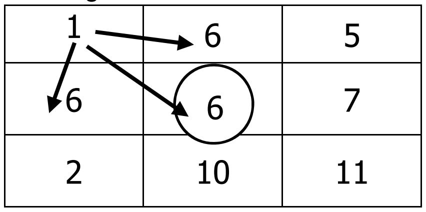
that returns cost of path to reach index (M-1, N-1) from (0, 0). Each cell of the matrix represents the cost of using that cell. The total cost of a path to reach (M-1, N-1) is the sum of cells used in the path including the source (0,0) and destination (M-1, N-1). At any given cell (i, j), you can only go down (i+1, j), right (i, j+1) or diagonal (i+1, j+1). You will move to the cell with the maximum cost. Your function should meet the following conditions:

- Choose the maximum value



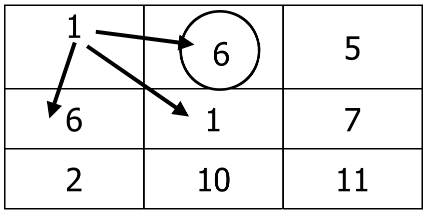
1	3	5
6	1	7
2	10	11

- If the diagonal *also* has the largest value, prioritize the diagonal one.



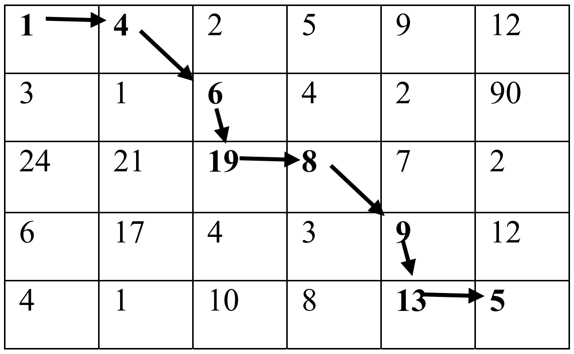
1	6	5
6	6	7
2	10	11

- If right and down values are equal and greater than diagonal, then choose the right.



1	6	5
6	1	7
2	10	11

Sample Example:



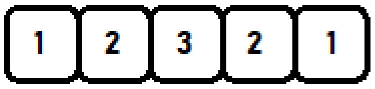
1	**4**	2	5	9	12
3	1	**6**	4	2	90
24	21	**19**	**8**	7	2
6	17	4	3	**9**	12
4	1	10	8	**13**	**5**

The function will return 65 since the cost = 1+4+6+19+8+9+13+5 = 65

Rough work:

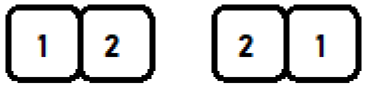
Solution Q2.

Q3. A student Laiba has a chocolate bar. The bar is n squares long. Each square has a number written on it. A sample bar is shown below: CLO-3



Ali wants a piece of chocolate, but Laiba decides she would only share the chocolate if a certain set of conditions are met. First, she will only share one connected portion and its size would be equal to Ali's birth month (m). Second, the sum of all numbers in this portion must be equal to Ali's birth day (d). You need to determine the number of ways Laiba can divide the chocolate.

Sample example: consider the chocolate bar given above with n=5. Suppose d = 3 and m = 2. Since m = 2, Ali will get a portion of size 2, and the sum of the two pieces must be 3 (d = 3). There are two ways Laiba can split this chocolate; from index 0 or from index 3 as shown below:



a) Implement the function **`void birthday`** (…) that takes in as input parameters a chocolate bar represented as an integer array s, and integers n, d and m for array size, Ali's birth day and birth month respectively. It also takes as an input parameter an integer array **`result`** that stores the initial index numbers of all legal splits followed by -1. For the example given above, the **`result`** would contain [0, 3, -1]. In case no split is possible, store -1 at **`result's`** index 0.

```
void birthday(________________________________________________)
{




}
```

b) Next, you need to implement the function `printPortions` that prints all possible legal portions that Laiba can share with Ali. The prototype is given below:

```
void print (int s[], int n, int m, int result[])
```

Using the same sample example as part (a):
Input: s = [1, 2, 3, 2, 1], n = 5, m = 2 and result = [0, 3, -1]
Print: (1,2) and (2,1).

Sample example 2:
Input: s = [1, 1, 1, 1, 1, 1], n = 6, m = 2 and result = [-1]
Print: “Don’t Share”

```
void print (int s[], int n, int m, int result[])
{









}
```

c) Implement the main function with appropriate variables, input statements and function calls to the functions implemented in parts (a) and (b).

```
int main()
{





}
```

Rough work:

Q4. The function `printData` takes in a NULL terminated char array '`data`' as an input parameter. This array contains details of different students like their names and marks for exactly 3 quizzes. The array begins with the first student's name followed by 3 marks. Names and marks are separated by #. The same pattern follows for the second student and so on. The records of different students are separated by the $ (dollar) sign. The function `printData` prints the name and average marks of all the students according to the pattern given in the sample example. The quiz marks are whole numbers from 0 to 9. CLO-3

```
Input: "Ali Khan#5#6#1$Saad Nawaz Khan#3#4#9$Rehmat Ali#3#4#5$"
Output:
Ali Khan 4
Saad Nawaz Khan 5.33
Rehmat Ali 4
```

Q5 marks 5*6 = 30

a. **Write the output of the following:** CLO-2

```
int main() {
    const int R = 3;
    const int C = 4;
    int myVals[R][C];
    int myVector[C] = { 4, 3, 2, 1 };
    for (int i = 0; i < R * C; i++) {
        myVals[i / C][i % C] = myVector[i % C]+1;
        cout << myVals[i / C][i % C] << "\t";
        if ((i % C) == (C - 1)) cout << "\n";
    }
    return 0;
}
```

Q5b. Initialize a 1D integer array in the main function and print its contents on odd indices in reverse order without using if statement. CLO-1

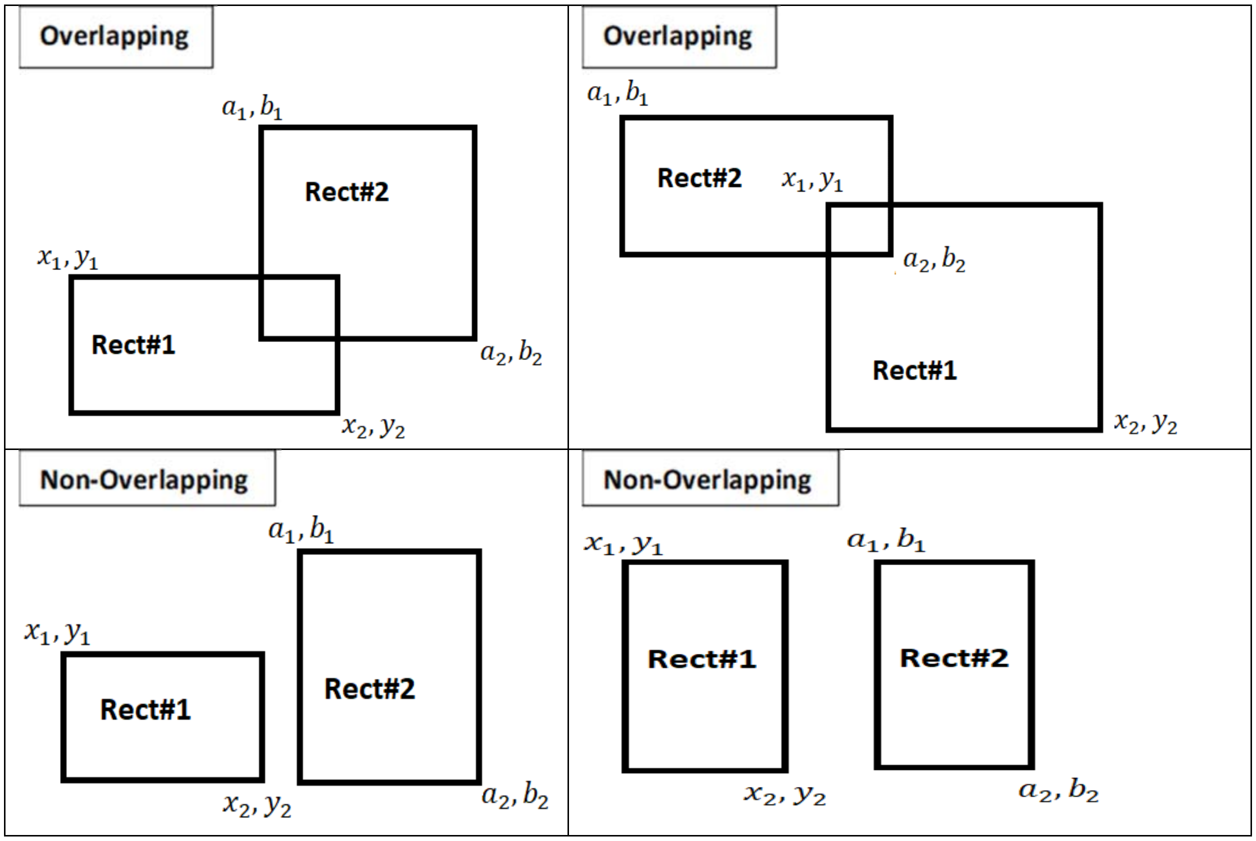
Q5c. Take a number as input and print ‘yes’, if it is even and divisible by both 3 and 5. CLO-1

Q5d. Read 10 numbers from a file named “data.txt”, and print their sum on screen. CLO-1

Q5e. Assume that we have two rectangles namely (Rect#1, and Rect#2) and we want to determine whether these rectangles are overlapping or not. The coordinates of upper left corner and lower right corners of both rectangles are given. Where (x_1, y_1) and (x_2, y_2) are the upper left corner and lower right corner coordinates of rect#1 and similarly (a_1, b_1) and (a_2, b_2) are the upper left corner and lower right corner coordinates of rect#2 respectively. Your task is to write a program that prompts the user to enter the value of these coordinates. (Assume that both the rectangles lie in the first quadrant i.e., all the values are positive). CLO-1

Determine and display the message whether the rectangles are overlapping or not. **YOU CAN ONLY USE THE IF statement/clause only ONCE.** Switch and loops are not allowed.

Sample examples:



```
int main(){
     int x1, y1, x2, y2, a1, b1, a2, b2;
     cin>> x1>> y1>> x2>> y2>> a1>> b1>> a2>> b2
```

Your code here

For rough work [turnover for next question]

Q5f. Implement the function `orderNamesByLength` that takes in exactly 3 char arrays as input parameters. CLO-1

```
void orderNamesByLength(char name1[], char name2[], char name3[]);
```

This function will sort them on the basis of the length of the names. `name1` should store the shortest name, while `name3` should store the longest one. The function `orderNamesByLength` should only do the work of rearranging or swapping the character arrays according to length. Assume the functions `Lenght` which returns the length of the char array and function `Copy` that copies the second array into the first one, are already implemented. You must use these functions in `orderNamesByLength`. Sample example:

Input: `name1` = "Junaid Ahmad", `name2` = "Ali M.", `name3` = "Ali Khan"

After function call: `name1` = "Ali M.", `name2` = "Ali Khan", `name3` = "Junaid Ahmad"