National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	Programming Fundamentals	Course Code:	CS1002
Program:	BS (CS, DS, SE)	Semester	Fall 2023
Duration:	180 Minutes	Total Marks:	90
Paper Date:	23 December 2023	Page(s):	12
Section	All Sections	CLOs	2,3,3,2,1,1
Exam Type:	Final Exam		

Student Name: **Section:**

Registration #:

Instructions
Carefully write your answer(s) in the space provided on this paper.
USE ROUGH SHEETS ONLY FOR WORKING
WRITE YOUR ANSWER IN THE SPACE PROVIDED ON THIS PAPER**.**
The term **EXPLANATION** used in this exam is to explain the sample example and should not be part/output of the program.
Good luck!

Question No 1. [10 Points]

Provide complete definition of the function **longestSubarray** that finds the longest sequence of 1's that can be obtained after ignoring <u>exactly one</u> of the elements of the array passed to it. You can assume that the array contains only zeros and/or ones.

Example 1:	Example 2:	Example 3:
Input: nums = [1,1,0,1] **Output:** 3 **Explanation:** After ignoring the number in position 2, we get 3 numbers with value of 1's.	**Input:** nums = [0,1,1,1,0,1,1,0,1] **Output:** 5 **Explanation:** The longest sequence of 1's that we get is 5 after ignoring the number in index 4,	**Input:** nums = [1,1,1] **Output:** 2 **Explanation:** You must ignore one element.

```cpp
int longestSubarray(int nums[], int size) {
    int maxLength = 0;
    int left = 0;
    int right = 0;
    int zeroCount = 0;

    while (right < size) {
        if (nums[right] == 0) {
            zeroCount++;
        }

        while (zeroCount > 1) {
            if (nums[left] == 0) {
                zeroCount--;
            }
            left++;
        }

        maxLength = std::max(maxLength, right - left + 1);

        right++;
    }
```

```
    return maxLength - 1;  // Subtracting 1 because we're allowed to delete one
element
}




















}
```

Question No 2.   [20 Points]

Encryption is a process of encoding information in such a way that only authorized parties can access it. It is a primary way of protecting sensitive data from unauthorized access, disclosure, or alteration. Some encryption technique that take a simple text in ASCII, usually called the plaintext, and convert it into a disguised form, called ciphertext, using an integer called a key of encryption. Following is description of one such algorithm/technique.

For example, suppose you want to encrypt a plaintext consisting of **n** letters $t_1, t_2, t_3, \ldots, t_n$ by using an integer key **k** such that **k>=1**. Then the letters in the ciphertext are t_1, t_{k+1}, t_{2k+1} **.. until you reach of end of plain text**. Starting the cycle again, the next characters in the ciphertext are $t_2, t_{k+2}, t_{2k+2}, \ldots$ **until the end of plaintext** is reached, then repeat the cycle from $t_3, t_{k+3}, t_{2k+3} \ldots$ and we keep repeating till $t_k, t_{k+k}, t_{2k+k}, \ldots$

Consider the sample example for the plaintext "MY.NAME.IS.DON" and k = 2.
Cycle 1: The letters in the ciphertext should be $t_1, t_{2+1}, t_{2*2+1} \ldots$, i.e., M.AEI.O
Cycle 2: The next characters in the ciphertext should be $t_2, t_{2+2}, t_{2*2+2} \ldots$, i.e., YNM.SDN
Stop as k=2.

The cipher text becomes: M.AEI.OYNM.SDN

Some more sample examples are shown in the following table that gives the ciphertext produced for the plaintext "MY.NAME.IS.DON" using various values of key.
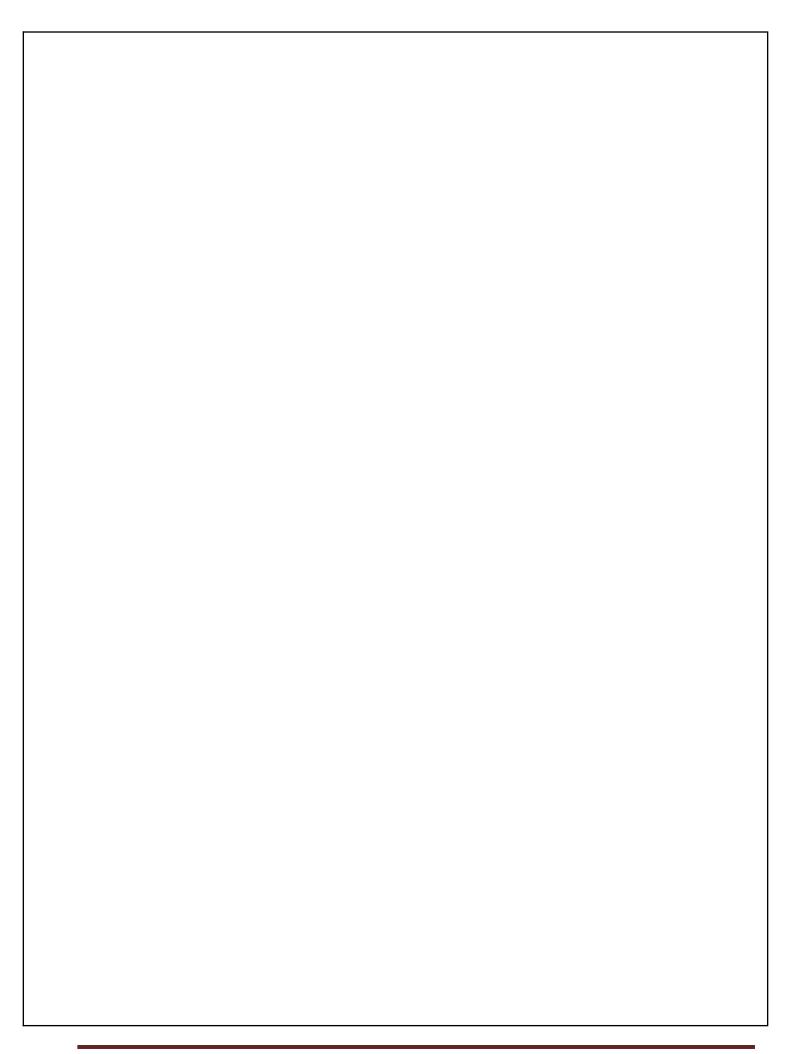
Key	Ciphertext
2	M.AEI.OYNM.SDN

3	MNESOYA..N.MID
5	MM.YED..ONINAS
20	MY.NAME.IS.DON
1	MY.NAME.IS.DON

QUESTION: Your job is to write a function named **encryptIT** that takes a null terminated character array containing plaintext, an integer key and return the ciphertext to the calling function as a null terminated string using a character array argument also passed to this function.

Note: You might find the length of a c-string using the strlen function available in the cstring library. Furthermore you might use built-in functions but please clearly mention the purpose of any such function used in your code

```cpp
#include <iostream>
#include <cstring>

using namespace std;

void encryptIT(char plaintext[], int key, char ciphertext[]) {
    // Get the length of the plaintext
    size_t length = strlen(plaintext);

    // Initialize the index for the ciphertext array
    size_t resultIndex = 0;

    // Iterate through each cycle
    for (int i = 0; i < key; ++i) {
        for (size_t j = i; j < length; j += key) {
            // Append the current character to the ciphertext
            ciphertext[resultIndex++] = plaintext[j];
        }
    }

    // Null-terminate the ciphertext
    ciphertext[resultIndex] = '\0';
}

int main() {
    char plaintext[100] = "MY.NAME.IS.DON";
    int key = 5;

    // Calculate the length of the ciphertext using the provided formula
    size_t length = strlen(plaintext) + (strlen(plaintext) - 1) / key;

    // Declare a fixed-size array for the ciphertext
    char ciphertext[length + 1];

    // Encrypt the plaintext
    encryptIT(plaintext, key, ciphertext);

    // Display the result
    cout << "Key " << key << ": " << ciphertext << endl;

    return 0;
}
```

Question No 3. [20 Points]

An automatic transport system has been installed in various cities of a developed country and in every city it has one operational bus that has a capacity of 100 passengers at any given time. Each time the bus starts at a different starting point and moves towards the city railway station that is at right-bottom location of the city. The map of the city is a square grid with each crossing indicating a bus station. The information about the number of passengers waiting to go to the railway station at each station are saved in a 2D array. A value of
-1 in this array indicates that the station is under construction and neither a passenger nor a bus can go that way. Map of a city of size 4 is shown below

City Map

0	1	0	-1
0	2	0	7
2	5	0	0
0	3	2	0

The auto-driver software controlling the bus use the following very simple logic to drive it
- A bus always moves right if possible and down only if the right move is not possible either because the station is under construction or we have reached the end of city.

Write two functions named

i)   **inputMap** that can be used to input values in the city map from the user. The Map and size of city must be arguments of this function.

    AND

ii)   **countPassengers** that takes as argument the city Map (assume that the maximum size of any city is defined as a global constant CITY_MAX), the size of the city, and the starting row and column position of the bus in the city. The function must return the total number of passengers transported to station by that bus. If the bus cannot reach the station using the simple logic because both the right station and down station are under construction then your function must return -1

You can assume that at the maximum possible number of passengers enter into the bus at each station and no passenger leaves before reaching the station. Your function must work perfectly with the following main

```
int main(){
    int Map[CITY_MAX][CITY_MAX],  citySize, sRow, sColumn;
        cout<<"Enter Map Size ";
        cin>> citySize;
        InputMap(Map, citySize);
        cout<<"Enter Starting Row Position and Column Position Of Bus ";
        cin>> sRow>>  sColumn;
```

```cpp
        while(0 <= sRow && sRow < citySize && 0 <= sColumn && sColumn < citySize){
            cout<<" Maximum Passengers: ";
            cout<< countPassengers(Map, citySize, sRow, sColumn)<<endl;
            cout<<"Enter Starting Row Position and Column Position Of Bus ";
            cin>> sRow>> sColumn;
        }
}
```

HINT: creating some functions to find valid move might simplify the logic of core function.

Following table gives are some sample inputs and corresponding outputs

Sample Input/Output Example 1	Sample Input/Output Example 2
City Size = 4 Input Map of city	City Size = 3 Input Map of city

Sample Input/Output Example 1:

City Size = 4
Input Map of city

0	1	0	-1
0	2	0	7
2	5	0	0
0	3	2	0

Starting position	Passenger count
Index [0][0]	8
Index [2][0]	7
[-1] [0]	Program Terminates

Explanation:
In the first case Bus goes R-R-D-R-D-D
In the second case Bus goes R-R-R-D

Sample Input/Output Example 2:

City Size = 3
Input Map of city

0	1	-1
0	1	0
2	-1	0

Starting position	Passenger count
[0] [0]	2
[2] [0]	-1
[1] [0]	1
[-1] [-1]	Program Terminates

Explanation:
In the first case Bus goes R-D-R-D
In the second case Bus cannot move R or D
in the third case Bus goes R-R-D

```cpp
#include <iostream>
using namespace std;
const int CITY_MAX = 10;  // Adjust the maximum city size as needed

void inputMap(int Map[CITY_MAX][CITY_MAX], int citySize) {
   cout << "Enter the city map (" << citySize << "x" << citySize << "):" << endl;
   for (int i = 0; i < citySize; ++i) {
     for (int j = 0; j < citySize; ++j) {
       cout << "Map[" << i << "][" << j << "]: ";
       cin >> Map[i][j];
     }
   }
}
```

```cpp
int countPassengers(int Map[CITY_MAX][CITY_MAX], int citySize, int sRow, int sColumn)
{
    int totalPassengers = 0;
    int row = sRow, col = sColumn;

    while (0 <= row && row < citySize && 0 <= col && col < citySize) {
        totalPassengers += Map[row][col];

        // Check if the bus can move right or down
        if (col + 1 < citySize && Map[row][col + 1] != -1) {
            col += 1;  // Move right
        } else if (row + 1 < citySize && Map[row + 1][col] != -1) {
            row += 1;  // Move down
        } else {
            break; // Bus cannot reach the station
        }
    }

    if (row == sRow && col == sColumn) {
        return -1;  // Bus cannot reach the station
    } else {
        return totalPassengers;
    }
}

int main() {
    int Map[CITY_MAX][CITY_MAX], citySize, sRow, sColumn;

    cout << "Enter Map Size: ";
    cin >> citySize;

    inputMap(Map, citySize);

    cout << "Enter Starting Row Position and Column Position Of Bus: ";
    cin >> sRow >> sColumn;

    while (0 <= sRow && sRow < citySize && 0 <= sColumn && sColumn < citySize) {
        cout << "Maximum Passengers: " << countPassengers(Map, citySize, sRow, sColumn) << endl;
        cout << "Enter Starting Row Position and Column Position Of Bus: ";
        cin >> sRow >> sColumn;
    }

    return 0;
}
```

Question No 4. [20 Points]

SOFTEC, Software Exhibition and Competition, is an annual event held at the National University of Computer and Emerging Sciences Lahore, Pakistan. Several guest from industry and academia come to attend SOFTEC in order to find talented future industry leaders and researchers. FAST admin maintains a log of entry and exit timings of the guests. This log is in the form of an unordered list of arrival and exit times and might contain invalid entries. **Any entry with either the arrival or exit time outside the range of 9:00 am (i.e. 9 hours) to 10 pm (i.e. 22 hours) are invalid entries.**

We need to find the **time** at which **maximum number of guests** are available in the SOFTEC and the maximum number of guest at that time. You can exclude those exiting at that time. In order to store the entry and exit time of guest, two arrays `arrival` and `exit` are maintained. As an example, `arrival[0]` stores the arrival time of the first guest, while `exit[0]` has the exit time of the same guest. A partial implementation of a program that performs this tasks is given below. Some of the functions in this partial implementation are completely missing whereas some have missing code. Complete the implementation of the all necessary functions to make the program fully functional. Ignore include statements.

```cpp
Question 4


#include<iostream>
using namespace std;

const int MAX_GUESTS = 100;
const int MAX_TIME_RANGE = 14; // Range from 9 to 22 (10 - 9 + 1)

void InputGuest(int arrival[], int exit[], int numGuests) {
    cout << "Enter the entry and exit times for each guest:" << endl;
    for (int i = 0; i < numGuests; ++i) {
        cout << "Guest " << (i + 1) << " entry time: ";
        cin >> arrival[i];

        cout << "Guest " << (i + 1) << " exit time: ";
        cin >> exit[i];
    }
}

bool isValidEntry(int entryTime, int exitTime) {
    return (entryTime >= 9 && entryTime <= 22 && exitTime >= 9 && exitTime <= 22);
}

void findMaxGuestsTime(int arrival[], int exit[], int numGuests, int& maxGuests, int& maxTime) {
    if (numGuests == 0) {
        // Case where there are no guests
        maxGuests = -1;
        maxTime = -1;
        return;
    }

    // Find min and max time within the valid range
```

```cpp
    int minTime = 9;
     maxTime = 22;

    // Create a count array with a fixed size
    int count[MAX_TIME_RANGE] = { 0 };

    // Traverse intervals and update count array for valid entries
    for (int i = 0; i < numGuests; ++i) {
        if (isValidEntry(arrival[i], exit[i])) {
            for (int j = max(arrival[i], minTime); j <= min(exit[i], maxTime); ++j) {
                count[j - minTime]++;
            }
        }
    }

    // Find the index of the maximum element in count array
    int maxIndex = 0;
    for (int i = 1; i < MAX_TIME_RANGE; ++i) {
        if (count[i] > count[maxIndex]) {
            maxIndex = i;
        }
    }

    // Set the result values
    maxGuests = count[maxIndex];
    maxTime = maxIndex + minTime;
}

int main() {
    int arrival[MAX_GUESTS], exit[MAX_GUESTS];
    int numGuests, maxGuests, maxTime;

    cout << "Enter the number of guests: ";
    cin >> numGuests;

    InputGuest(arrival, exit, numGuests);

    findMaxGuestsTime(arrival, exit, numGuests, maxGuests, maxTime);

    if (maxGuests == -1 || maxGuests==0) {
        cout << "There are no guests." << endl;
    }
    else {
        cout << "Time with maximum guests: " << maxTime << " with " << maxGuests << " guests." << endl;
    }

    return 0;
}
```

Two sample inputs and the corresponding output are given in the following table

Sample Inputs:	Outputs
Input: Number of guests 5 Input: arrival [] = 9 11 15 13 19 Input: exit [] = 16 20 22 21 22	The time with maximum guests is: 15 with 4 guests.
Input: Number of guests 6 Input: arrival [] = 6 11 15 13 19 22 Input: exit [] = 16 23 22 14 21 22	The time with maximum guests is: 19 with 2 guests.

Question No 5.   **[5 Points]**

Dry run the following code and write is output in the space provided.

```cpp
#include <iostream>
using namespace std;
int main() {
        int amount = 0;
        for (int i = 1; i <= 2; i++)
                for (int j = 1; j <= 4; j++) {
                        if (j == 2)
                                continue;
                        for (int k = 1; k <= 4; k++) {
                                if (k == 3)
                                        break;
                                if (k != i && k != j && i != j) {
                                        amount++;
                                        cout << i << j << k << " "<<endl;
                                }
                        }
                }
        cout << "Total numbers: " << amount << endl;
}
```

132
142
231
241
Total numbers: 4

Question No 6. [15 Points]

In the space provided on the right side, write the exact output produced by each part of the **main** function given below? Indicate no output by writing the symbol ☹.

You must assume that the following functions are already defined and no include or using

```cpp
#include<iostream>
using namespace std;

char fun(int x){
    return x;
}
char Double(int x){
    return x*2;
}
int chill(int x,int y){
    int m;
    if(x < y) m=x; else m=y;
    for(int i=m;i;i--)
        if(x%i==0 && y%i==0)
            return i;
    return -1;
}
```

Part		OUTPUT

a)	`int main ()`
	`{`
	`    int i = 5, j = 6, k = 7, n = 3;`
	`    float x;`
	`    char l = 'A';`
	`    int arr[5]={1,2,3};`
	`    cout << k - i * k + k % n << endl;`
		-27

b)	`n=0, k=1;`
	`if (!n)`
	`    cout << "danger" << endl;`
		danger

c)	`if (--k == 0)` `    cout << "No danger" << endl;`	No danger
d)	`if (n || k)` `    cout << "Nothing" << endl;`	☹
		G

e)	`cout << fun(i+j+Double(i)*j) << endl;`	

f)	`char ch, field[] = "Computer";`
	`ch = field [1];`
	`field [3] = ch;`
	`cout << field << endl;`
		Comouter

`cout << fun(i+j+Double(i)*j) << endl;`

`char ch, field[] = "Computer";`
`ch = field [1];`

PART		OUTPUT
| g) | ```
n=k=5;
while (n > 0)
    n /= 2;
cout << n * n << endl;
``` | 0 |
| h) | ```
while (l <= 'Z'){
    cout << l << endl;
    l = l + 12;
}
``` | A<br>M<br>Y |
| i) | ```
cout << chill(200,40) << endl;
``` | 40 |
| j) | ```
cout << chill(17,4) << endl;
``` | 1 |
| k) | ```
int& b = n;
b = 5;
cout << n;
``` | 5 |
| l) | ```
bool f = true, t = false;
i = 10;
j = 5;
n = ((f||t) + (i + j));
cout << n;
``` | 16 |
| m) | ```
cout << arr[4] << endl;
``` | 0 |

```
/*
For Question (n) and (o), assume the given functions are implemented and
provide the output of the particular piece of code.
swap function swaps 2 number.
max3 returns the largest of the three numbers
space prints n (n as input parameter) consecutive spaces in a line
plus prints n (n as input parameter) consecutive plus signs in a line
*/
```

PART		OUTPUT
| n) | ```
swap(arr[4], n);
swap(i, n);
swap(i, arr[3]);
cout << arr[4] << n << i << endl;
``` | **16**<u>10</u>*0* |
| o) | ```
plus(max3(arr[1], n, c-120+22));
spaces(arr[4]-10);
plus(1);
``` | |

```
        return 0;
}
```