


National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Programming Fundamentals	Course Code:	CS 118
	Program:	BS(CS,SE,DS)	Semester:	Fall 2020
	Duration:	180 Minutes	Total Points:	65
	Paper Date:	11-02-2021	Page(s):	13
	Exam Type:	Final	Section:	ALL
	Section:	ALL		

Roll Number: _____

Instructions:

Attempt all questions

You might use extra sheets for working but do not attach them.

Write the final answer in the space provided for it.

Question No.	Marks	Marks Obtained
Q1	10	
Q2	10	
Q3	15	
Q4	15	
Q5	15	
Total	65	

Question No. 1:**[5 x 2 Points]**

What would be the output of the following code?	Output
<pre> int WHAT(int A[], int N){ int ANS = 0; int S = 0; int E = N-1; for(S = 0, E = N-1; S < E; S++, E--) ANS += A[S] - A[E]; return ANS; } int main(){ int A[] = {1, 2, 3, 4, -5, 1, 3, 2, 1}; cout<< WHAT(A, 7); return 0; } </pre>	7
Write the output for one of the following programs.	
<pre> void Sum(int a) { cout << a + 100 << endl; } void Sum(int a, int b, int c = 10) { cout << a + b + c << endl; } int main() { Sum('A'); Sum('B', 30); Sum(20, 30, 90.5); return 0; } </pre>	165 106 140
<pre> struct Std { int ID; char Grade; double PFMarks; }; int main() { Std X; int ID = 1; int PFMarks = 0; X.ID = ID; X.Grade = 'B'; X.PFMarks = 70.5; cout << ID << X.Grade << X.PFMarks << endl; return 0; } </pre>	1B70.5

Question No. 2:**[2 x 5 Points]**

Identify and describe the syntax errors in the following code snippets and suggest how could we correct them.

Code	Error description and Correction
<pre>int main() { cout << (rand() / 5.3 + 20) % 10; return 0; }</pre>	<p>Description: Left operand of % has type double</p> <p>Correction: cout << (rand() / 5 + 20) % 10;</p>
<pre>void fun(int & value) { value++; cout << value; } int main() { int x = 10; fun(x + 30); return 0; }</pre>	<p>Description: The function parameter is a reference, but it can't refer to x+30 because x+30 is a value and not a variable</p> <p>Correction: fun(x);</p>
<pre>void changeMe(const int x, int y) { y = x - 30 / 100; x = y + 50; } int main() { int x = 100; x = changeMe(x, 200); cout << x; return 0; }</pre>	<p>Description: 1. Cannot modify a const variable 2. A value of type of "void" cannot be assigned to an entity of type "int"</p> <p>Correction: void changeMe(const int x, int y) { y = x - 30 / 100; } int main() { int x = 100; changeMe(x, 200); cout << x; return 0; }</p>
<pre>int main() { int arr[2][3] = {{1, 2}, {3, 4}, {5,6}}; return 0; }</pre>	<p>Description: Array size and number of initializer values don't match</p> <p>Correction: int arr[3][2] = {{1, 2}, {3, 4}, {5, 6}}; OR int arr[2][3] = {{1, 2, 3}, {4, 5, 6}};</p>

```

void input (int arr[][], int row, int col)
{
    for (int i = 0; i < col; i++) {
        for (int j = 0; j < row; j++)
            cin >> arr[j][i];
    }
}

int main() {
    int arr[3][2];
    input (arr, 3, 2);
    return 0;
}

```

Description:

Number of columns for the parameter array is missing

Correction:

```
void input(int arr[][2], int row, int col)
```

Question No. 3:**[15 Points]**

Consider the following algorithm to generate a sequence of numbers.

1. Start with a positive integer n .
2. If n is even, divide by 2.
3. If n is odd, multiply by 3 and add 1.
4. Repeat this process with the new value of n , terminating when n becomes 1.

For example, the following sequence of numbers will be generated for $n = 22$:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

It is conjectured (but not yet proven) that this algorithm will terminate at $n = 1$ for every integer n . Still, the conjecture holds for all integers up to at least 1,000,000.

For an **input n** , the **cycle-length** of n is the number of numbers generated up to and including the **1**. In the example above, the **cycle length** of **22** is **16**.

Similarly, the **cycle length** of **8** is **4** because the sequence **8 4 2 1** will be generated when we start at 8.

Write a C++ function that takes an array **A** of integer numbers as parameter and it **returns the cycle length of every number stored in A** using a second array **B** which is also passed to the function as a parameter.

Following is a suggested prototype of the function.

void cycleLengths (int A [], int B [], int N)

Here A and B are two arrays with the first N indices of A containing the numbers.

HINTS:

1. Making an additional function that computes the cycle length of a single integer and then calling it from function **CycleLengths** might simplify the code structure
2. You should not use any input or output statements for this problem
3. No need to write the main function

BLANK PAGE FOR WRITING ANSWER

```
int computeCycleLength(int n) {
    int length = 1;

    while (n != 1) {
        if (n % 2)
            n = n * 3 + 1;
        else
            n /= 2;
        ++length;
    }
    return length;
}

void cycleLengths(int A[], int B[], int N) {
    for (int i = 0; i < N; ++i)
        B[i] = computeCycleLength(A[i]);
}
```

BLANK PAGE FOR WRITING ANSWER

Question No. 4:**[15 Points]**

Write the C++ code for a function **Mirror-TwoD** which takes a two-dimensional integer array with its rows and column size as input parameters. This function verifies and returns true, if the data elements in array creates a mirror effect, and false otherwise.

A Mirror array:

1. Has the same number of rows and columns, (Square)
2. Has the same values on the main diagonal
3. The data elements and their ordering are same above and below main diagonal

For Example, the arrays given below are Mirror-TwoD

1	1	1
1	1	1
1	1	1

1	2	3
2	1	6
3	6	1

1	2	3	4
2	1	6	7
3	6	1	9
4	7	9	1

5	2	2	2
2	5	2	2
2	2	5	2
2	2	2	5

-1	1	2	-3
1	-1	2	4
2	2	-1	5
-3	4	5	-1

But the following arrays are not Mirror-TwoD

1	2	3
2	1	2
1	2	3

1	2	3	4
2	1	6	7
3	6	1	9
4	5	9	1

5	2	2	8
2	5	2	2
8	2	5	2
2	5	2	5

3	1	1	1
1	3	1	1
1	1	4	1
1	1	1	3

1	2	3	4
2	1	6	7
3	6	1	9

BLANK PAGE FOR WORKING AND WRITING ANSWER

```
bool Mirror_TwoD(int arr[][4], int rows, int cols) {
    if (rows != cols)
        return false;

    for (int i = 0; i < rows; ++i) {
        if (arr[0][0] != arr[i][i])
            return false;
        for (int j = i + 1; j < cols; ++j)
            if (arr[i][j] != arr[j][i])
                return false;
    }
    return true;
}
```

BLANK PAGE FOR WRITING ANSWER

Question No. 5:**[15 Points]**

Roman numerals are a numeral system that originated in ancient Rome. In this system the symbols **I, V, X, L, C, D, and M**, are used for representation of integer values **1, 5, 10, 50, 100, 500, and 1,000** respectively, in Arabic-numerals system. The rules for generating numbers are as follows:

1. Read the symbols from right to left.

Example: VIII 

2. When a symbol appears before a larger (or equal symbol) then symbol's value is added.

Example: VII = 5 + 1 + 1 = 7

Example: LXXX = 50 + 10 + 10 + 10 = 80

Example: DCLXVI = 500 + 100 + 50 + 10 + 5 + 1 = 666

3. If a smaller symbol appears after a larger symbol then its value is subtracted

Example: IV = V – I = 5 – 1 = 4

Example: IX = X – I = 10 – 1 = 9

Example: CM = 1000 – 100 = 900

4. Do not use the same symbol more than three times in a row

Example: Instead of IIII = 4, write IV = 5 – 1 = 4

Example: Instead of CCCC = 400, write DC = 500 - 100 = 400

- A. Convert the following numbers from Roman Numerals to corresponding integers in Arabic-numerals.
[2 Points]

Numbers in Roman Numerals	Integer Number
LXIV	64
MCCXXXIV	1234
MMMCCCXXXIII	3333
MMMDCLXVI	3666

- B. Consider that the following program code is already written, for conversion of the Roman Numerals to integer numbers.

```
#include <iostream>

using namespace std;

//Global arrays to store the Roman-Numeral Symbols and their corresponding values.
char symbol[7] = { 'I', 'V', 'X', 'L', 'C', 'D', 'M'};
int  value[7] = { 1, 5, 10, 50, 100, 500, 1000};

int findIndex(char ch);

int romanToArabic(char romanNum[]);

int main(){
    char romanNum [20] = "";

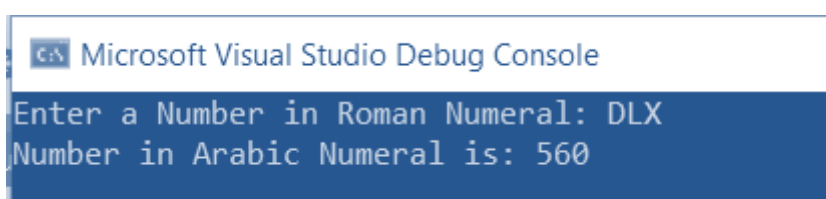
    cout << "Enter a Number in Roman Numeral: ";

    cin >> romanNum; //Input the roman numeral in a chracter array from user.

    cout << "Number in Arabic Numeral is: " << romanToArabic (romanNum) << endl;

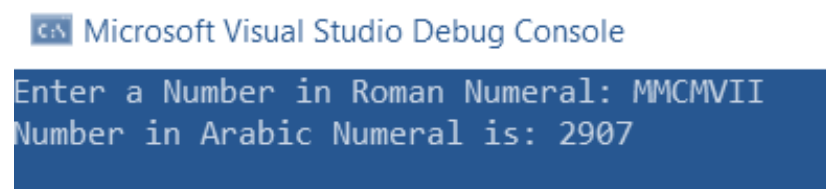
    return 0;
}
```

Following are the sample outputs:



Microsoft Visual Studio Debug Console

```
Enter a Number in Roman Numeral: DLX
Number in Arabic Numeral is: 560
```



Microsoft Visual Studio Debug Console

```
Enter a Number in Roman Numeral: MMCMVII
Number in Arabic Numeral is: 2907
```

Write the C++ code for the function `int findIndex(char ch);` which takes a character as **input parameter** and returns the index of that character, if the given character is found in **symbol** array, and -1 otherwise. [5 Points]

```
int findIndex(char ch) {  
    for (int i = 0; i < 7; ++i)  
        if (symbol[i] == ch)  
            return i;  
    return -1;  
}
```

C. Write the C++ code for the function `int romanToArabic(char romanNum[]);` which takes a null terminated character array, containing the number in Roman Numeral, as input parameter, it converts this number to corresponding integer number and returns the converted integer. [8 Points]

1. You can assume that the input Roman Numeral number is always correct.
2. If needed, you can use the built-in functions to manipulate strings using any of the available options or built-in functions.
3. You **must** use the global array **value** and **findindex** function to compute the value

BLANK PAGE FOR WRITING ANSWER

```
int romanToArabic(char romanNum[]) {
    int intNumber = 0;
    int i = strlen(romanNum) - 1;

    while (i >= 1) {
        int index1 = findIndex(romanNum[i]);
        int index2 = findIndex(romanNum[i - 1]);

        if (index1 <= index2) {
            intNumber += value[index1];
            --i;
        }
        else {
            intNumber = value[index1] - value[index2];
            i -= 2;
        }
    }

    if (i == 0)
        intNumber += value[findIndex(romanNum[0])];

    return intNumber;
}

// Function to convert the roman to integer number in Arabic Numeral
int romanToArabic(char romanNum[]) {
    int arabicNum = 0;
    int length = strlen(romanNum);
    int index1 = 0;
    for (int i = length-1; i >=0; i--) {
        index1 = findIndex(romanNum[i]);
        if ((i<length-1) && (index1 < findIndex(romanNum[i+1])))
            arabicNum -= value[index1];
        else
            arabicNum += value[index1];
    }
    return arabicNum;
}
```

BLANK PAGE FOR WRITING ANSWER

