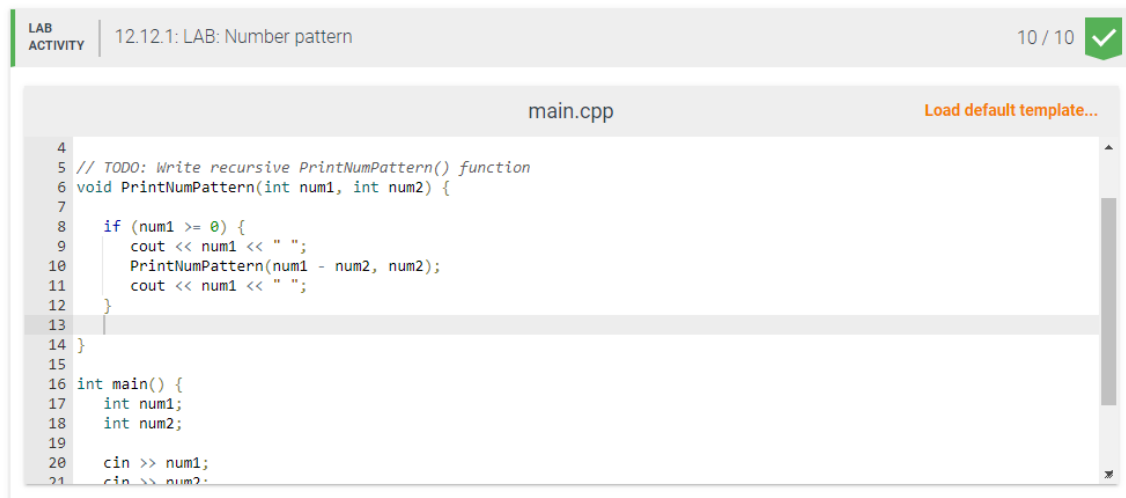



CSCI 140 PA 11 Submission

Due Date: 5/23/23

Name: Ali Mortada

Exercise 1 – **zyBook 12.12 LAB: Number pattern** -- check if completely done in zyBook __X__ ; otherwise, discuss issues below
Include a screenshot of current status/score



LAB ACTIVITY | 12.12.1: LAB: Number pattern | 10 / 10 

main.cpp [Load default template...](#)

```
4
5 // TODO: Write recursive PrintNumPattern() function
6 void PrintNumPattern(int num1, int num2) {
7
8     if (num1 >= 0) {
9         cout << num1 << " ";
10        PrintNumPattern(num1 - num2, num2);
11        cout << num1 << " ";
12    }
13
14 }
15
16 int main() {
17     int num1;
18     int num2;
19
20     cin >> num1;
21     cin >> num2;
```

Exercise 2 – **zyBook 12.14 LAB: Count the digits** -- check if completely done in zyBook ____ ; otherwise, discuss issues below
Include a screenshot of current status/score



LAB ACTIVITY | 12.14.1: LAB: Count the digits | 8 / 10 

main.cpp [Load default template...](#)

```
1 #include <iostream>
2 using namespace std;
3
4
5 /* TODO: Write recursive DigitCount() function here. */
6 int digitCount = 0;
7 int DigitCount(int num) {
8
9     if (num == 0) {
10        return digitCount;
11    } else {
12        digitCount++;
13        DigitCount(num / 10);
14    }
15 }
16
17
```

Exercise 3 – **Recursive Find in an Array** -- check if completely done __X__ ;
otherwise, discuss issues below

Source code below:

```
/*
    Program: Recursive Find in an Array
    Author: Ali Mortada
    Class: CSCI 140
    Date: 5/23/23
    Description: Uses a recursive function to find the location of first found element
in an array. Returns location of element if found or -1 if element is not found.
    I certify that the code below is my own work.
    Exception(s): N/A
*/

#include <iostream>
using namespace std;

int recFind(const int a[], int key, int start, int n) {

    // If value is not found, return -1
    if (n - start == 0) {
        return -1;
    }
    // If value is equal to key, return index of found value
    else if (a[start + 1] == key) {
        return start + 1;
    }
    // Else, recursively call function to keep searching
    return recFind(a, key, start + 1, n);
}

int main() {

    cout << "Author: Ali Mortada" << endl << endl;

    int values[7] = {5, 7, 3, 2, 6, 2, 15};

    cout << "Index of 2 : " << recFind(values, 2, 0, 7) << endl;    // 3
    cout << "Index of 1 : " << recFind(values, 1, 0, 7) << endl;    // -1
    cout << "Index of 15: " << recFind(values, 15, 0, 7) << endl;    // 6

    return 0;
}
```

Input/output below:

```
Author: Ali Mortada  
  
Index of 2 : 3  
Index of 1 : -1  
Index of 15: 6
```

Exercise 4 – **Recursive Array Reverser** -- check if completely done __X__ ; otherwise, discuss issues below

Source code below:

```
/*  
    Program: Recursive Find in an Array  
    Author: Ali Mortada  
    Class: CSCI 140  
    Date: 5/23/23  
    Description: Uses a recursive function to reverse the elements of an int array.  
    I certify that the code below is my own work.  
    Exception(s): N/A  
*/  
  
#include <iostream>  
using namespace std;  
  
void recArrReverse(int a[], int start, int end) {  
  
    // Once array has been iterated through, exit the function  
    if (start >= end) {  
        return;  
    } else {  
        // Swap first and last values  
        int temp = a[start];  
        a[start] = a[end];  
        a[end] = temp;  
  
        // Recursively call function to swap second/second-last, third/third-last, etc.  
        recArrReverse(a, start + 1, end - 1);  
    }  
}  
  
// Function to print arrays  
void printArray(int a[], int size) {  
  
    for (int i = 0; i < size; i++) {
```

```

        cout << a[i] << " ";
    }

    cout << endl;
}

int main() {

    cout << "Author: Ali Mortada" << endl << endl;

    int values[7] = {5, 7, 3, 2, 6, 2, 15};
    cout << "Regular array: ";
    printArray(values, 7); // 5 7 3 2 6 2 15
    cout << endl;

    recArrReverse(values, 0, 6);
    cout << "Reversed array: ";
    printArray(values, 7); // 15 2 6 2 3 7 5

    return 0;
}

```

Input/output below:

```

Author: Ali Mortada

Regular array: 5 7 3 2 6 2 15

Reversed array: 15 2 6 2 3 7 5

```

Answer for Question 1

A recursive solution is not necessary to solve the above problems, since both can be solved using loops. For the recursive find function, one could write a loop that iterates through the array and compares each value to the key, then return the index of the first instance of the key in the array. For the recursive array reverser, one could write a loop where the first and last values are switched, then the second and second-last values are switched, etc. until the array is iterated through.

Answer for Question 2

One advantage of recursion is that it can reduce time complexity for certain problems, especially those based on tree structures. However, recursion uses lots of stack memory, and can cause a stack overflow error when dealing with a large data set/used too many times.

Extra Credit – provide if applicable.

LAB
ACTIVITY

12.10.1: LAB: All permutations of names

10 / 10

main.cpp

Load default template...

```
1 #include <vector>
2 #include <string>
3 #include <iostream>
4
5 using namespace std;
6
7 // TODO: Write function to create and output all permutations of the list of names.
8 void PrintAllPermutations(vector<string> &permList, vector<string> &nameList) {
9     vector<string> tempList;
10
11     if (nameList.size() == 0) {
12         for (int i = 0; i < permList.size(); i++) {
13             if (i != permList.size() - 1) {
14                 cout << permList.at(i) << ", ";
15             } else {
```