**CMPSC122 In-Lab 2 - Review of C++ Arrays**

**Part I. In-Lab Exercise (**10 points)

Submit the solution files of Exercises online before the due date/time. Upload your file named InLab2.docx. You can use this handout and append your answer in between, please use a different color font for your answers for easy grading. Example:

1. A pointer variable stores a(n) memory address. <type your answer here or in a separate line,

separate answers with a comma if any>

Here are some concepts in C++ for review:   
Basic Abstract Data Types: int, bool, float, double, char   
Arrays and loops

**Exercise 1.** (2 points) Examine the C++ program below, and answer the question about that program in the blanks provided.  
  
#include<iostream>  
using namespace std;

int main()  
{  
    const int N = 20; // why const? N is declared as a constant because its value remains constant throughout the program, indicating the size of arrays  
    int I, J, K, Sum;  
    float A[10];  
    double B[100];  
    char C[N+1];  
    short D[] = {1, 3, 5, 7, 9};  
    int E[N];  
  
 // The operator "sizeof" evaluates the number of bytes for its operand  
    cout << "Size of A: " << sizeof( A ) << endl; // Size of A: 40  
    cout << "Size of B: " << sizeof( B ) << endl; // Size of B: 800

    cout << "Size of C: " << sizeof( C ) << endl; // Size of C: 21  
    cout << "Size of D: " << sizeof( D ) << endl; // Size of D: 10  
    cout << "Size of E: " << sizeof( E ) << endl; // Size of E: 80  
    K = sizeof( D ) / sizeof( D[0] );  
    cout << "Elements in D: " << K << endl; // Elements in D: 5  
    cout << "D[2]: " << D[2] << endl; // D[2]:5  
    cout << "D[4]: " << D[4] << endl; // D[4]:9  
    for (I=0; I<K; I++)  
    {  
      D[I] = D[I] \* 2 + 1;  
    }  
    cout << "Last I: " << I << endl; // Last I:5

    cout << "D[2]: " << D[2] << endl; // D[2]:11

    cout << "D[4]: " << D[4] << endl; // D[4]:19

    for (I=0; I<N; I++)  
    {  
       E[I] = N - I;  
    }  
    cout << "E[9]: " << E[9] << endl; // E[9]:11

    cout << "E[12]: " << E[12] << endl; // E[12]:8

    cout << "E[15]: " << E[15] << endl; // E[15]:5  
    for (I=0; I<N; I++)  
    {  
        E[I] = I;  
    }  
    for (I=9; I>=0; I--)  
    {  
        Sum = 0;  
        for (J=0; J<I; J++)  
        {  
            Sum += E[J];  
        }  
        E[I] = Sum;  
    }  
    cout << "E[5]: " << E[5] << endl; // E[5]: 10  
    cout << "E[7]: " << E[7] << endl; // E[7]:21  
    cout << "E[9]: " << E[9] << endl; // E[9]:36  
    E[0] = 1;   
    E[1] = 1;  
    for (I=2; I<N; I++)  
    {  
        E[I] = E[I-1] + E[I-2];  
    }  
    cout << "E[4]: " << E[4] << endl; // E[4]:5  
    cout << "E[5]: " << E[5] << endl; // E[5]:8

    cout << "E[6]: " << E[6] << endl; // E[6]:13  
    cout << "E[9]: " << E[9] << endl; // E[9]:55

}

**Exercise 2.** (2 points) Examine the C++ program below, and predict the value displayed by the program for each output operation. Please try to fill in the blanks with your knowledge before running these codes using the C++ compiler.  
  
  
#include<iostream>  
using namespace std;  
  
// A function that takes two int pointers p and q as parameters   
void SomeFunc(int \* p, int \* q) {  
 int temp = \*p;  
 \*p = \*q;  
 \*q = temp;  
}  
  
// A function that takes two int pointers p and q and an int reference n as parameters  
void Func2(int \* p, int \* q, int &n) {  
 \*p = \*q + n;  
 \*q = (\*p) \* n;  
 \*p = \*q - n;  
}

int main() {  
 int \*ptr1,\*ptr2;  
 int a,b;  
 a = 4;  
 b = 3;  
 ptr1 = &a;  
 ptr2 = ptr1;  
 \*ptr2 = 9;  
 cout << a << endl; //\_\_\_\_\_\_9\_\_\_\_  
 ptr2 = &b;  
 \*ptr2 = \*ptr1;  
 \*ptr1 = 12;  
 cout << b << endl; //\_\_\_\_\_9\_\_\_\_\_  
  SomeFunc(ptr1, ptr2);  
 cout << a << endl; //\_\_\_\_\_9\_\_\_\_\_\_\_\_  
 cout << b << endl; //\_\_\_\_\_\_12\_\_\_\_\_\_  
 \*ptr1 = \*ptr2;

if(ptr1 == ptr2) {  
 cout << "Equal!" << endl;  
 }  
 else {  
    cout << "Not Equal!" << endl;  
 } // \_\_\_Not Equal!\_\_

a = 5;  
 b = 6;  
  Func2(ptr1,ptr2,a);  
 cout << \*ptr1 << endl; //\_\_\_\_\_110\_\_\_\_\_  
 cout << \*ptr2 << endl; //\_\_\_\_\_121\_\_\_\_\_  
}

**Exercise 3**. (2 points) The following exercises are designed to give you practice with C++ pointers. For each of the following, write a single statement that performs the indicated task.  Assume that floating-point variables number1 and number2 have been declared and that number1 has been initialized to 7.3.

1. Declare the variable fPtr to be a pointer to an object of type double.

double\* fPtr;

1. Assign the address of the variable number1 to the pointer variable fPtr.

fPtr = &number1;

1. Print the value of the object pointed to by fPtr.

cout << \*fPtr;

1. Assign the value of the object pointed to by fPtr to the variable number2.

number2 = \*fPtr;

1. Print the value of the number2.

cout << number2;

1. Print the address of the number1.

cout << &number1;

1. Print the address stored in fPtr. Is the value printed the same as the address of the number1?

cout << fPtr;

Yes it should be the same because we assigned the address of number1 to fPtr in task2

**Exercise 4.** (6 points) Write a program (Upload your file named TestScores.cpp) that dynamically allocates an array large enough to hold a user-defined number of test scores. Once all the scores are entered, the array should be passed to a function (use your preferred sorting function) that sorts them in ascending order. Another function should be called that calculates the average score. The program should display the sorted list of scores and averages with appropriate headings. Use pointer notation rather than array notation whenever possible. Finally, drop the lowest test score and compute the new average score using a new array dynamically allocation.

Input Validation: Do not accept negative numbers for test scores.

Sample Run:

Please enter the number of test scores: 0

Invalid input. Please try again.

Please enter the number of test scores: 6

Enter test score 1: 69

Enter test score 2: 93

Enter test score 3: -78

Invalid input: negative value. Please try again.

Enter test score 3: 75

Enter test score 4: 96

Enter test score 5: 91

Enter test score 6: 88

Sorted test scores:

===================

69

75

88

91

93

96

The average score: 85.33

Sorted test scores (lowest dropped):

====================================

75

88

91

93

96

Average test score after dropping lowest score: 88.60