Lab 12

LAB 12.1 Introduction to Pointer Variables

Use the code below:

```
// This program demonstrates the use of pointer variables
// It finds the area of a rectangle given length and width
// It prints the length and width in ascending order

// PLACE YOUR NAME HERE

#include <iostream>
using namespace std;

int main()
{
    int length;          // holds length
  int width;  // holds width
  int area;    // holds area

  int *lengthPtr = nullptr;       // int pointer which will be set to point to length
  int *widthPtr = nullptr;        // int pointer which will be set to point to width

    cout << "Please input the length of the rectangle" << endl;
  cin >> length;

    cout << "Please input the width of the rectangle" << endl;
  cin >> width;

    // Fill in code to make lengthPtr point to length (hold its address)

    // Fill in code to make widthPtr point to width (hold its address)

    area =    // Fill in code to find the area by using only the pointer variables

    cout << "The area is " << area << endl;

    if (// Fill in the condition length > width by using only the pointer variables)
      cout << "The length is greater than the width" << endl;

    else if (// Fill in the other condition by using only the pointer variables)
      cout << "The width is greater than the length" << endl;

    else
      cout << "The width and length are the same" << endl;

    return 0;
}
```

Exercise 1

Complete this program by filling in the code (places in bold).

Note: use only pointer variables when instructed to by the comments in bold. This program is to test your knowledge of pointer variables and the & and * symbols.

Exercise 2

Run the program with the following data: 10 15.

Record the output here: _____

LAB 12.2 Dynamic Memory

Use the code below:

```
// This program demonstrates the use of dynamic variables

// PLACE YOUR NAME HERE

#include <iostream>
using namespace std;

const int MAXNAME = 10;

int main()
{
   int pos;
   char *name = nullptr;
   int *one = nullptr;
   int *two = nullptr;
   int *three = nullptr;
   int result;

   //    Fill in code to allocate the integer variable one here

   //    Fill in code to allocate the integer variable two here

   //    Fill in code to allocate the integer variable three here

   //    Fill in code to allocate the character array pointed to by name


   cout << "Enter your last name with exactly 10 characters." << endl;
   cout << "If your name has < 10 characters, repeat last letter. " << endl
        << "Blanks at the end do not count." << endl;

   for (pos = 0; pos < MAXNAME; pos++)
         cin >>      // Fill in code to read a character into the name array
               // WITHOUT USING a bracketed subscript

   cout << "Hi ";

   for (pos = 0; pos < MAXNAME; pos++)
         cout << // Fill in code to a print a character from the name array
                 // WITHOUT USING a bracketed subscript

   cout << endl << "Enter three integer numbers separated by blanks" << endl;

   // Fill in code to input three numbers and store them in the
   // dynamic variables pointed to by pointers one, two, and three.
   // You are working only with pointer variables
```

```
    // echo print
    cout << "The three numbers are " << endl;

    // Fill in code to output those numbers


    result = // Fill in code to calculate the sum of the three numbers

    cout << "The sum of the three values is " << result << endl;

    // Fill in code to deallocate one, two, three and name


    return 0;

}
```

Exercise 1

Complete the program by filling in the code (areas in bold). Study the code carefully in order to complete the program. The expected output (with example input underlined) is given below.

Sample Run:

```
Enter your last name with exactly 10 characters.
If your name < 10 characters, repeat last letter. Blanks do not count.
Kevinnnnnn
Hi Kevinnnnnn
Enter three integer numbers separated by blanks
5 6 7
The three numbers are 5 6 7
The sum of the three numbers is 18
```

Exercise 2

In inputting and outputting the name, you were asked NOT to use a bracketed subscript. Why is a bracketed subscript unnecessary?

Would using `name[pos]` work for inputting the name? Why or why not?
Would using `name[pos]` work for outputting the name? Why or why not?

Try them both and see.

LAB 12.3 Dynamic Arrays

Use the code below:

```
// This program demonstrates the use of dynamic arrays

// PLACE YOUR NAME HERE

#include <iostream>
#include <iomanip>
using namespace std;

int main(){
   float *monthSales = nullptr;  // a pointer used to point to an array
                                 // holding monthly sales

   float total = 0;  // total of all sales
   float average;          // average of monthly sales
   int numOfSales;         // number of sales to be processed
   int count;              // loop counter

   cout << fixed << showpoint << setprecision(2);

   cout << "How many monthly sales will be processed? ";
   cin >> numOfSales;

   // Fill in the code to allocate memory for the array pointed to by
   // monthSales.

   if ( // Fill in the condition to determine if memory has been
        // allocated (or eliminate this if construct if your instructor
        // tells you it is not needed for your compiler) )
   {
         cout << "Error allocating memory!\n";
         return 1;
   }

   cout << "Enter the sales below\n";

   for (count = 0; count < numOfSales; count++){
         cout << "Sales for Month number     "
               << // Fill in code to show the number of the month
               << ":";

         // Fill in code to bring sales into an element of the array

   }

   for (count = 0; count < numOfSales; count++)
   {
         total = total + monthSales[count];
   }

   average = // Fill in code to find the average

   cout << "Average Monthly sale is $" << average << endl;

   // Fill in the code to deallocate memory assigned to the array.

   return 0;
}
```

Exercise 1
Fill in the code as indicated by the comments in bold. Example user inputs are underlined.

Sample Run:

```
How many monthly sales will be processed: 3
Enter the sales below
Sales for Month number 1: 401.25
Sales for Month number 2: 352.89
Sales for Month number 3: 375.05
Average monthly sale is $376.40
```

LAB 12.4 Student Generated Code Assignments

In these assignments you are asked to develop functions that have dynamic arrays as parameters. Remember that dynamic arrays are accessed by a pointer variable and thus the parameters that serve as dynamic arrays are, in fact pointer variables.

Example:

```
void sort(float* score, int num_scores); // a prototype whose function has a
// dynamic array as its first
// parameter. It is a pointer variable
.
int main()
{
float *score = nullptr; // a pointer variable
.
.
score = new float[num_scores]; // allocation of the array

sort(score,scoreSize); // call to the function
```

Option 1:

Write a program that will read scores into an array. The size of the array should be input by the user (dynamic array). The program will find and print out the average of the scores. It will also call a function that will sort (using a bubble sort) the scores in ascending order. The values are then printed in this sorted order.

Sample Run (user inputs are underlined):

```
Please input the number of scores
5
Please enter a score
100
Please enter a score
90
Please enter a score
95
Please enter a score
100
Please enter a score
90
The average of the scores is 95
```

```
Here are the scores in ascending order
90
90
95
100
100
```

Option 2:

This program will read in id numbers and place them in an array. The array is dynamically allocated large enough to hold the number of id numbers given by the user. The program will then input an id and call a function to search for that id in the array. It will print whether the id is in the array or not.

Sample Run (user inputs are underlined):

```
Please input the number of id numbers to be read
4
Please enter a score
96
Please enter a score
97
Please enter a score
98
Please enter a score
99

Please input an id number to be searched
67
67 is not in the array
```

Exercise 3

Write a program that will read monthly sales into a dynamically allocated array. The program will input the size of the array from the user. It will call a function that will find the yearly sum (the sum of all the sales). It will also call another function that will find the average.

Sample Run (user inputs are underlined):

```
Please input the number of monthly sales to be input
4
Please input the sales for month 1
1290.89
Please input the sales for month 2
905.95
Please input the sales for month 3
1567.98
Please input the sales for month 4
994.83
The total sales for the year is $4759.65
The average monthly sale is $1189.91
```