CMPT 125: Week 2 Lab Work

1. What is the output of the following program?

```
#include <iostream>
using namespace std;
int main()
{
        char a, b, c;
        a = 'b';
        b = 'c';
        c = a;
        cout << a << b << c << 'c' << endl;

        system("Pause");
        return 0;
}
```

2. What is the output of the following C++ code fragments assuming it is placed inside a valid C++ program?

```
int a = 5, b = 7;
cout << ++a << ", " << b++ << endl;
cout << a-- << ", " << ++b << endl;
cout << ++a << ", " << --b << endl;
cout << a-- << ", " << b++ << endl;
cout << ++a + b++ << endl;
cout << a-- + ++b << endl;
int c = a++ + ++b;
cout << c << endl;
c = a-- - --b;
cout << c << endl;
c = c++ - ++a + --b;
cout << c << endl;
```

3. What is the output of the following code snapshot assuming it is embedded inside a valid C++ program

```
int a = 25;
int b = 4;
cout << a / b << endl;
cout << static_cast<float>(a) / b << endl;
cout << a / static_cast<float>(b) << endl;
cout << static_cast<float>(a) / static_cast<float>(b) << endl;
cout << static_cast<float>(a / b) << endl;
cout << a  << ", " << b << endl;
float result = a / b;
cout << result << endl;
result = static_cast<float>(a / b);
cout << result << endl;
int c = static_cast<float>(a) / b;
cout << c << endl;
c = a / static_cast<float>(b);
cout << c << endl;
result = static_cast<float>(a) / static_cast<float>(b);
c = static_cast<float>(a) / static_cast<float>(b);
cout << result << ", " << c << endl;
```

4. Analyze the following program and determine its output

```
#include <iostream>
using namespace std;
int main()
{
        char c1 = 65;
        char c2 = 321;
        char c3 = -191;
        char c4 = 'A';
        cout << c1 << endl;
        cout << c2 << endl;
        cout << c3 << endl;
        cout << c4 << endl;
        system("Pause");
        return 0;
}
```

5. What is the output of the following program?

```
#include <iostream>
using namespace std;
int main()
{
        char c1 = 'A';
        char c2 = c1 + 5;
        cout << c1 << endl;
        cout << c2 << endl;
        system("Pause");
        return 0;
}
```

6. What is the output of the following programs?

`int n = 0; while (n++ < 5) { cout << ++n << endl; } cout << n << endl;`	`int n = 0; while (++n < 5) { cout << n++ << endl; } cout << n << endl;`	`int n = 0; while (n++ < 5) { cout << n++ << endl; } cout << n << endl;`
`int n = 0; while (++n < 5) { cout << ++n << endl; } cout << n << endl;`	`int n = 0; while (n+1 < 5) { cout << n++ << endl; } cout << n << endl;`	`int n = 0; while (++n < 5) { cout << n+1 << endl; } cout << n << endl;`

7. Write a complete C++ program that declares six variables named y1, m1, d1, y2, m2, and d2 all as integer data types. Now read the birth date of a child1 in the d1, m1 and y1 variables where d1, m1 and y1 represent the day, month and year of the birth date of child1. Then read the birth date of child2 in d2, m2, and y2 variables. Finally print the number of days ***BETWEEN*** the birth dates of the two children. Assume each month has 30 days and each year has 12 months (= 360 days).

For example, if d1 = 18, m1 = 4, y1 = 1998 and d2 = 10, m2 = 9, y2 = 1995 then your program must print **There are 938 days between the birth dates.**

As another example, if d1 = 10, m1 = 9, y1 = 1995 and d2 = 18, m2 = 4, y2 = 1998 then your program must still print **There are 938 days between the birth dates.**

8. Write a complete C++ program that declares six variables named y1, m1, d1, y2, m2, and d2 all as integer data types. Now read the birth date of a child1 in the d1, m1 and y1 variables. Similarly, read the birth date of child2 in d2, m2, and y2 variables. Finally print the number of years, number of months and number of days ***BETWEEN*** the birth dates of the two children. Assume each month has 30 days and each year has 12 months (= 360 days).

For example, if d1 = 18, m1 = 4, y1 = 1998 and d2 = 10, m2 = 9, y2 = 1995 then your program must print **There are 2 years, 7 months, and 8 days between the birth dates.**

As another example, if d1 = 10, m1 = 9, y1 = 1995 and d2 = 18, m2 = 4, y2 = 1998 then your program must still print **There are 2 years, 7 months, and 8 days between the birth dates.**

9. **[Challenge]** Write a C++ program that prints **randomly** one of the following three messages: **Yes**, **No** or **Not-Sure**.

10. Write a program that reads a positive integer number **n** and then prints the double data type value **y** given by:

$$y = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \cdots \pm \frac{1}{n}$$

Note that the last term may have a plus sign or a minus sign depending on the value of **n**. Use a **while** loop. Assume that the user input value for **n** is positive integer. Also note that the divisions must be performed in a float/double domain to avoid truncation.

11. Define a C++ function named **`isDigit`** that takes a character argument and returns true if the argument is a digit and returns false otherwise. Write also a test program to test main program your function.

12. Define a function named **`printIncreasingOrder`** that takes three float arguments and prints them in increasing order. What does your function return? Write also a test program to test main program your function.

13. Define a function named **`reversedInteger`** that takes one integer argument and returns an integer made up of the digits of the argument in reverse order. Write also a test program to test main program your function. Example,

`reversedInteger(65)` must return 56
`reversedInteger(0)` must return 0
`reversedInteger(-762)` must return -267

14. Define a C++ function named **`printPrimes`** that takes one integer argument **n** and prints all the primes in the range [2, n]. Write also a test program to test main program your function.

15. Define a function named **`allEven`** that takes one positive integer argument **n**. Your function then must generate **n** random integers in the range [-20, 20], prints each of the random numbers generated, and finally return true if all the randomly generated integers are even numbers and returns false otherwise. Write also a test program to test main program your function.

16. Define a function named **`allPrimes`** that takes one positive integer argument **n**. Your function then must generate **n** random integers in the range [2, 100], prints each of the random numbers generated, and finally returns true if all the randomly generated integers are prime numbers and returns false otherwise. Write also a test program to test main program your function.

17. Define a function named **`quadraticTester`** that takes three float arguments **a**, **b**, and **c** and that returns the number of real solutions (int data type) of the quadratic equation given by **$ax^2+bx+c=0$.** Write also a test program to test main program your function.

18. Define a function named **`allIncreasing`** that takes one positive integer argument **n** and that generates **n** random integers in the range [1, 100], prints each of the random numbers generated, and finally returns

true if the generated random integers appeared in increasing order; otherwise returns false. Write also a test program to test main program your function.

19. Define a function named **`getLetter`** that takes one upper case English letter character argument and that returns the English letter that is ten letters away from the argument. For example,

If the argument character is 'A' then your function must return 'K'
If the argument character is 'B' then your function must return 'L'
If the argument character is 'C' then your function must return 'M'
If the argument character is 'Q' then your function must return 'A'
If the argument character is 'V' then your function must return 'F'
If the argument character is 'Z' then your function must return 'J'

Observe that if the character that is ten letters away from the argument is outside of the upper case English letters; then we must go back to 'A'. Write also a test program to test main program your function.

20. Given two points on a plane P1(x1, y1) and P2(x2, y2); the length of the line segment connecting P1 and P2 is given by $\boldsymbol{d} = \sqrt{(\boldsymbol{x1} - \boldsymbol{x2})^2 + (\boldsymbol{y1} - \boldsymbol{y2})^2}$. Write a C++ function named **`vectorLength`** that takes four arguments x1, y1, x2, and y2 and returns the length of the line segment connecting the two points P1 and P2. Write also a test program to test main program your function.

21. Consider the area of a triangle whose three vertices on the plane are given as points P1(x1,y1), P2(x2,y2),and P3(x3,y3). Define a C++ function named **`triangleArea`** that takes six arguments x1, y1, x2, y2, x3, and y3 and returns the area of the triangle. Assume no two points are collinear. **Hint:-** First calculate the length of each of the sides of the triangle using the Euclidean Distance formula. For this you should make use of your function **`vectorLength`**. Write also a test program to test main program your function.

22. Given the cubic polynomial function

$$f(x) = ax^3 + bx^2 + cx + d$$

Define a C++ function named **`f`** that takes five double data type arguments **a**, **b**, **c**, **d**, and **x** where **a**, **b**, **c**, and **d** are the coefficients of the cubic polynomial and **x** is a point whose ***f(x)*** we would like to compute, and returns the value of ***f(x)*** as a double data type. In order to make it a general C++ function that works for any cubic order polynomial; this function must take all the coefficients **a**, **b**, **c**, and **d** as arguments. In addition, the function has to take the value of **x** as an argument. The function returns ***f(x)***. The declaration of the function and the testing main program is given below in order to get you started.

```
float f(const float a, const float b, const float c, const float d, const float x)
{
        //Put function body here
}
int main()
{
        float a, b, c, d;
        cout << "Enter the coefficients of the cubic polynomial: ";
        cin >> a >> b >> c >> d;
        //Now let us find f(x) for ten different values of x
        float x;
        for (int i = 0; i < 10; i++)
        {
                cout << "Enter value of x ";
                cin >> x;
                cout << "The value of f(" << x << ") = " << f(a,b,c,d,x) << endl;
        }
        system("Pause");
        return 0;
}
```

23. [Optional! Not required for the course] **Roots of a cubic polynomial function**:- Given the cubic polynomial function

$$f(x) = ax^3 + bx^2 + cx + d$$

Define a C++ function named `polyRoot` that takes four double data type arguments **a**, **b**, **c**, and **d** which are the coefficients of the cubic polynomial and such that $a \neq 0$ and that returns a root of f (that is, a value of x such that $f(x) = 0$) as a double data type.

Please note that given an odd degree polynomial function such as a cubic polynomial, it is guaranteed that the graph of the function crosses the x-axis at least once. This means there exists at least one value of x such that $f(x) = 0$. Of course there may be more than one values of x such that $f(x) = 0$ in which case your function can return any one of them.

One way to compute a root of an odd degree polynomial will be as follows. First of all recall that a function $f(x)$ has the same roots as the function $-f(x)$. With this in mind, if the coefficient $a < 0$, then multiply each of the coefficients a, b, c, and d by -1 so that the coefficient a > 0 and thus the function becomes an increasing function. Then perform the following steps:

Step 1. Select two initial guesses $x1$ and $x2$ such that $f(x1) < 0$ and $f(x2) > 0$. Since we have made sure that $a > 0$ and thus f is an increasing function, then the choice of $x1$ and $x2$ can be achieved by selecting some big negative value for $x1$ and some big positive value for $x2$. Now there exists an x in **[x1, x2]** such that $f(x) = 0$.

Step 2. Compute the middle value $x = (x1+x2)/2$.

Step 3. Compute $f(x)$ and

- If $f(x) = 0$ then we are done we have found a required solution.
- Else if $f(x) > 0$ then there has to be a solution in the range **[*x1, x*]**. Therefore we make the update by assigning $x2$ the value of x and go to Step 2 above.
- Else (that is if $f(x) < 0$) then there has to be a solution in the range **[*x, x2*]**. Therefore we make the update by assigning $x1$ the value of x and go to Step 2 above.

Repeat this process until you find a middle value x such that $f(x) = 0$.

Remark 1. The easiest way to compute the initial guesses x1 and x2 is to initialize x1 = -1.0 and x2 = 1.0 and then decrement x1 until $f(x1) < 0$ and increment x2 until $f(x2) > 0$.

Remark 2. In step 3 above, it is possible that $f(x)$ is never exactly equal to zero and therefore we may end up in an infinite loop. In order to avoid an infinite loop, we should stop the loop when $f(x)$ is very close to zero such as when $abs(f(x)) < 0.00001$. This will give us a solution correct to four decimal places which is great.

This method of computing a solution by repeatedly bisecting a given interval into two sub-intervals is known as the bisection method. It is a very easy to understand and powerful numerical method that can be used in many practical problems; albeit it is a very slow algorithm.

24. [Optional! Not required for the course] **Bisection method for computing square roots of non-negative numbers:-** Define a C++ function named `mySquareRoot` that takes a non-negative double data type argument a and returns the square root of a.

Hint:- Use the bisection method discussed above with $f(x) = x^2 - a$ and the initial guesses of $x1 = 0.0$ and $x2 = a + 1$. Below is the function declaration and a test main program to test your function.

```
double mySquareRoot(const double a)
{
     //Pre-condition: a is a non-negative double data type value
     //Post-condition: returns the square root of a
     //Algorithm: Bisection method
}
int main()
{
     double num;
     cout << "Enter a non-negative number ";
     cin >> num;
     double y = mySquareRoot(num);
     cout << "The square root of " << num << " is " << y << endl;
     system("Pause");
     return 0;
}
```