# CMPT 130: Lab Work Week 9

**1.** What is the output of the following program?

```
#include <iostream>
using namespace std;
void figure_me_out(int& x, int y, int& z)
{
	cout << "Entering function: " << x << ", " << y << ", " << z << endl;
	x = 1;
	y = 2;
	z = 3;
	cout << "Exiting function: " << x << ", " << y << ", " << z << endl;
}

int main()
{
	int a = 10, b = 20, c = 30;
	cout << "Main Program before function call: " << a << ", " << b << ", " << c << endl;
	figure_me_out(a, b, c);
	cout << "Main Program after function call: " << a << ", " << b << ", " << c << endl;
	return 0;
}
```

**2.** What is the output of the following code fragment when embedded in a valid C++ main program?

```
int a = 1, b = 2, c =3;
int *d = &a, *e = &b, *f = &c;
int &g = a;
int &h = b;
int &k = c;
*e = k;
e = &h;
*d = h;
cout << g << ", " << b << ", "  << *f << endl;
```

**3.** What is the output of the following program?

```
void magic(int &a, int *b, int c)
{
	c = a;
	*b = a + c;
	a = c * *b;
}

int main()
{
	int x = 6, y = 8, z = 10;
	cout << x << "   " << y << "   " << z << endl;
	magic(y, &x, z);
	cout << x << "   " << y << "   " << z << endl;
	system("Pause");
	return 0;
}
```

**4.** Let's repeat Q#3 above but now with different variable names. Is there any syntax error in this program? What is the output of the following program?

```
void magic(int &a, int *b, int c)
{
	c = a;
	*b = a + c;
	a = c * *b;
}

int main()
{
	int a = 6, b = 8, c = 10;
	cout << a << "   " << b << "   " << c << endl;
	magic(b, &a, c);
	cout << a << "   " << b << "   " << c << endl;
	system("Pause");
	return 0;
}
```

**5.** Consider the following program and determine its output

```
void foo(int &m, int n, int *x, int *y)
{
	x = y;
	*y = 7;
	m = 8;
	n = *x;
	cout << m << ", " << n << ", " << *x << ", " << *y << endl;
	return;
}

int main()
{
	int a = 3, b = 5;
	int *p = &a, *q = &b;
	cout << *p << ", " << *q << endl;
	foo(a, b, p, q);
	cout << a << ", " << b << endl;
	return 0;
}
```

**6.** What is the output of the following C++ program?

```
int main()
{
	int a = 6, b = 8;
	int *p = &a, *q = &b, **r = &p;
	*p = *q;
	**r = 10;
	cout << a << ", " << b << ", " << *p << ", " << *q << ", " << **r << endl;
	*q = 12;
	*p = **r;
	cout << a << ", " << b << ", " << *p << ", " << *q << ", " << **r << endl;
	return 0;
}
```

7. Consider the quadratic equation $ax^2 + bx + c = 0$. Our aim is to write a C++ function that takes the three coefficients **a**, **b**, and **c** as arguments and returns the solutions of the quadratic equation, if any. So for the time being let us think of the function header to look like:

**float quadraticSolver(float a, float b, float c)**

But then we will be facing two problems with this approach:

- How can we return two solutions if the quadratic equation has two solutions? ***Remember C++ functions can return only one item!***, and
- What do we return if the quadratic equation doesn't have any solution?

What can we do then?

***Do references and pointers come to your mind?***

Why don't we pass two additional arguments to the function, say **s1** and **s2**.

These arguments are passed by either reference or pointer and the solutions computed are assigned to them accordingly.

But then what do we assign s1 and s2 if the quadratic equation doesn't have any solution?

The simplest way around this will be to design our function to return one of the integers **0, 1, or 2** depending on the number of solutions the quadratic equation has.

This way, the main program will only need to test the return value to know how many solutions are computed and proceed accordingly.

Therefore the skeleton of our function for parameter passing by reference should look like as follows:

```
int quadraticSolver(float a, float b, float c, float &sol1, float &sol2)
{
        /*
                If the quadratic equation has no solution, then
                        Do not assign any value to the parameters sol1 and sol2
                        return 0;
                If the quadratic equation has ONE solution, then
                        Assign both parameters sol1 and sol2 the value of the solution computed.
                        return 1;
                If the quadratic equation has TWO solutions, then
                        Assign both parameters sol1 and sol2 the values of the solutions computed.
                        return 2;
        */
}
```

Similarly, the same function with parameter passing by pointer will be

## int quadraticSolver(float a, float b, float c, float *sol1, float *sol2)

```
{
	.....
}
```

Now the following test main program can be used to test the function and show its usage.

```
int main()
{
	float a, b, c;
	cout << "Enter the coefficients a, b, c: ";
	cin >> a >> b >> c;
	float s1, s2;

	//Test parameter passing by reference
	int n = quadraticSolver(a, b, c, s1, s2);
	if (n==0)
		cout << "The quadratic equation has no solution" << endl;
	else if (n==1)
		cout << "The quadratic equation has one solution" << s1 << endl;
	else
		cout << "The quadratic equation has two solutions" << s1 << " and " << s2 << endl;

	//Test parameter passing by pointer
	n = quadraticSolver(a, b, c, &s1, &s2);
	if (n==0)
		cout << "The quadratic equation has no solution" << endl;
	else if (n==1)
		cout << "The quadratic equation has one solution" << s1 << endl;
	else
		cout << "The quadratic equation has two solutions" << s1 << " and " << s2 << endl;

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

Implement the **quadraticSolver** function in two different ways: one using parameter passing by reference as shown above and another one using parameter passing by pointers.

**8.** Consider the following program and determine its output. Also identify the line of code that must be removed from the program to avoid run time error.

```
int main()
{
	int *a = new int(7);//assume the heap memory has address 4F
	int *p;
	p = a;
	cout << a << endl;
	cout << p << endl;
	cout << *a << endl;
	cout << *p << endl;
	*p = 10;
	cout << *a << endl;
	delete p;
	delete a;
	return 0;
}
```

**9.** Consider the following program. Find the lines of code that must be removed from the main function or from the magic function; and also add some lines of code to the main function or the magic function so that the program runs correctly with no run-time error and that all the memory reserved in the heap will be cleared before the program ends.

```
int* magic()
{
	int *p = new int(7); //assume the heap memory has address 4F
	cout << p << endl;
	cout << *p << endl;
	delete p;
	cout << *p << endl;
	cout << p << endl;
	return p;
}
int main()
{
	int *a;
	a = magic();
	cout << a << endl;
	cout << *a << endl;
	delete a;
	*a = 6;
	cout << *a << endl;
	return 0;
}
```

**10.** Consider the following program. What is its output? What does the function foo return?

```
int& foo(int *p)
{
	*p = 7;
	return *p;
}

int main()
{
	int x = 5;
	int y = foo(&x);
	cout << x << ", " << y << endl;
	x = 12;
	cout << x << ", " << y << endl;
	y = 15;
	cout << x << ", " << y << endl;

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

**11.** What is the output of the following program?

```
int& magic(int &a, int *b, int c)
{
	c = a;
	*b = a + c;
	a = c * *b;
	return a;
}
int main()
{
	int x = 6, y = 8, z = 10;
	cout << x << "   " << y << "   " << z << endl;
	z = magic(y, &x, z);
	cout << x << "   " << y << "   " << z << endl;
	system("Pause");
	return 0;
}
```

**12.** What is the output of the following program?

```
int& foo(int *p)
{
	*p = 7;
	return *p;
}
int main()
{
	int x = 5;
	int& y = foo(&x);
	cout << x << ", " << y << endl;
	x = 12;
	cout << x << ", " << y << endl;
	y = 15;
	cout << x << ", " << y << endl;

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

**13.** What is the output of the following program?

```
int& foo(int *p)
{
	int* q = new int;
	*q = *p;
	return *q;
}
int main()
{
	int x = 5;
	int& y = foo(&x);
	cout << x << ", " << y << endl;
	x = 12;
	cout << x << ", " << y << endl;
	y = 15;
	cout << x << ", " << y << endl;
	delete &y;   //This will delete the heap memory created in foo function
	system("Pause");
	return 0;
}
```

**14.** Now consider the following program. If you type the code and run it, it may run fine. But it has a run time error. Explain the error.

```
int& foo(int *p)
{
	*p = 7;
	int q = *p;
	return q;
}

int main()
{
	int x = 5;
	int y = foo(&x);
	cout << x << ", " << y << endl;
	system("Pause");
	return 0;
}
```

**15.** Now consider the following program. If you type the code and run it, it may run fine. But it has a run time error. Explain the error.

```
int& foo(int *p)
{
	*p = 7;
	int q = *p;
	return q;
}
int main()
{
	int x = 5;
	int& y = foo(&x);
	cout << x << ", " << y << endl;

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

**16.** What is the output of the following program?

```
	int& foo(int *p)
	{
		*p = 7;
		return *p;
	}
	int main()
	{
		int x = 5;
		foo(&x) = 3;
		cout << x << endl;

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

**17.** What is the output of the following program?

```
int& magic(int &a, int *b, int c)
{
	c = a;
	*b = a + c;
	a = c * *b;
	return a;
}
int main()
{
	int x = 6, y = 8, z = 10;
	cout << x << "   " << y << "   " << z << endl;
	magic(y, &x, z) = 5;
	cout << x << "   " << y << "   " << z << endl;
	system("Pause");
	return 0;
}
```

**18.** What is the output of the following program?

```
int& magic(int &a, int *b, int c)
{
	c = a;
	*b = a + c;
	a = c * *b;
	return a;
}
int main()
{
	int x = 6, y = 8, z = 10;
	cout << x << "   " << y << "   " << z << endl;
	magic(y, &x, z) = z;
	cout << x << "   " << y << "   " << z << endl;
	system("Pause");
	return 0;
}
```