SENG2200/6220 –Programming Languages & Paradigms
Computer Lab for Week 3, Semester 1, 2020

1. Given the following section of C code, state where each of the variables is stored, when and where each is initialized, and what is the lifetime and visibility of each variable.

```
int a1 = 10, a2 = 20, a3 = 30;

void func1 (float f11, int f12) {
        float f13, f14 = 0.0;
        ….
}
int main( ) {
        int m1 = 0, m2 = 0, m3;
        func1 (m1, m2);
        ……
}
```

2. Given the following sections of C code, state where each of the variables is stored, when and where each is initialized, and what is the lifetime and visibility of each variable. Trace the execution of this program, showing what is stored in each variable as execution progresses.

```
int a1 = 10, a2 = 20, a3 = 30;

float  func2(int f21, int f22) {
        static int f23 = 40;
        static float f24 = 0.1;
        int f25 = 1; float f26;
        f26 = f24*f25+f22*f23/f21;
        f24 += 1.1;
        return f26;
}

int main( ) {
        float res1 = func2(a1, a2);
        float res2 = func2(a2, a3);
                …… /* assume results are output here
}
```

3. Describe a Stack using UML. Convert this description into a Java interface.

4. What is the problem with the following C++ code? How should it be fixed?

```
class Test {
    public:
        Test(int t);
        Test& newTest(int t);
        int getNum();
    protected:
        Test();
    private:
        int num;
};

//---

Test& Test::newTest(int num) {
    Test t(num);
    return t;
}
//---
```

5. Trace the memory usage for the following programs (ie what goes on the stack, what goes in the heap, etc.).

a.

```
public class Test {
  public static int count = 0;

  public void add(int c) {
    count += c;
  }

  public static void main(String[] args) {
    Test test = new Test();
    test.add(2);
    Test.count += 2;
  }
}
```

b.

```
class Test {
  public:
    static int count;
    void add(int c);
};


int Test::count = 0;
```

```
void Test::add(int c) {
  count += c;
}

int main() {
  Test test;
  test.add(2);
  Test::count += 2;
  ….
}
```

6.

a. Design (UML) and write a class (in Java) that will model a bank account. Each bank account must have an account number and a balance. Bank account numbers are simple integers but must be unique for each account. The balance cannot ever be negative. Valid operations on a bank account are deposit, withdraw, and getBalance (as well as whatever constructors are needed)
b. Now add data and methods that will allow the computation of "total deposits" and "total withdrawals" for this class.

7. Implement the above class in C++.