SENG2200 Week 3 Solutions

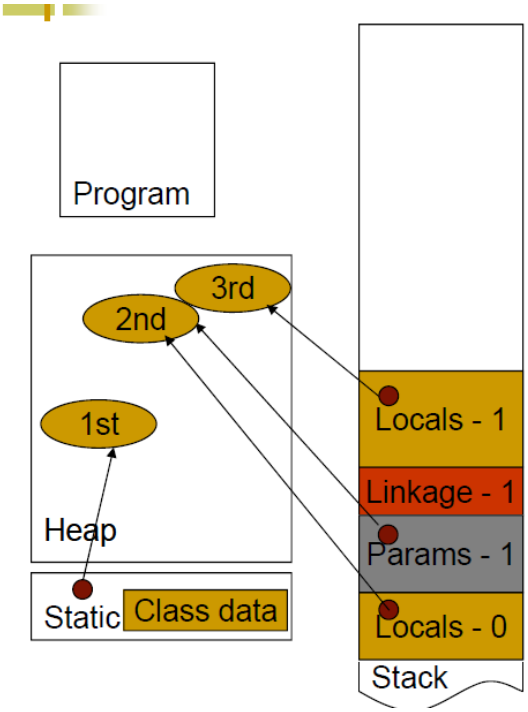
Q1)

int a1 = 10, a2=20, a3=30; this is visible everywhere and lasts for the entire program

f11, f12 (the params) are visible only in that function, exist only for the duration of that invocation of the function, and are stored in the 'params' section of the stack. A new version of them is created each time the function is called and they are initialised to the value passed in.

f13, f14 are stored in the 'locals' for the function func1, visible only by that function.

m1, m2, m3 are visible only inside **main** but exist for the life of the program (ie until **main** returns).



Q2)

Same as above for visibility, except that f23 and f24 cannot be seen outside of func2.

The value of f23 and f24 persist between calls to the function. They are initialised at program startup time, they simply do not become visible until the first call to the function and are then visible only during subsequent calls to the function. They are not re-initialised at each function call.

Make special note to the students about f23/f24 not being visible in main.

res1 = 0.1 * 1 + a2 * 40 / a1 = 0.1 + 800/10 = 80.1

res2 = 1.2 * 1 + a3 * 40 / a2 = 1.2 + 30*40/20 = 61.2

Q3)

Either an interface over a LL, or a class containing an array and a head pointer

O Pop(), push(O), O peek(), isEmpty()

Q4)

newTest creates an object on the stack in the method's local variable area, then returns a reference to it.

Once the function closes, that object is automatically destroyed so this leaves the returned reference dangling and not referring to anything useful.

To fix it, make it Test* t = new Test(num); which will put create the object on the heap, putting the object's lifetime under programmer control, and making sure it is still a valid object that can be referred to after the method has returned.

Q5)

count is in static section of memory

c, Args are on the stack

test is on the heap, but the reference to it is on the stack.

b)

count is static section of memory

test is an object on the stack local to main, c is on the stack local to add method.

6)

```
public class BankAccount  {

        private static int nextAccNo = 0;

        private int accNumber;
        private double balance;

        public BankAccount()  {
                Balance = 0;
                AccNumber =nextAccNo;
                nextAccNo++;
        }

        public void deposit(double amount)  {    // could return Boolean for success/fail
                If(amount < 0)
                        Throw new Exception();  // or just return – or set up to return true/false
                Balance += amount;
        }

        public void withdraw(double amount)  {    // could return boolean
                if(amount > 0 && balance >= amount)
                        balance -= amount;
                else
                        throw new Exception();  //  or as per above comments
        }

        public double getBalance()  {
                return balance;
        }
}
```

b) The same, but with static variables that have the amount always added to them on both withdraw and deposit (and static get methods for them, initialised to 0 in the class).

7)

```
class BankAccount  {

        private:
        static int nextAccNo = 0;
        int accNumber;
        double balance;

        public:
        BankAccount()
        void deposit(double amount)
        void withdraw(double amount)
        double getBalance()

};

        BankAccount::BankAccount()  {
                balance = 0;
                accNumber =nextAccNo;
                nextAccNo++;
        }

        void BankAccount::deposit(double amount)  {
                If(amount < 0)
                        Throw new Exception(); //should probably be returning false or similar

                Balance += amount;
        }

        void BankAccount::withdraw(double amount)  {
                if(amount > 0 && balance >= amount)
                        Balance -= amount;
                else
                        throw new Exception();//should probably be returning false or similar
        }

        doubleBankAccount:: getBalance()  {
                return balance;
         }
```