Com S 229 Fall 2011
Problem Set 3 (10 pts)
Due at 10:59am
Friday, October 28
Name: RYAN K SANDERS
ID (4-digit) : 8612

1. (2 points) Run the following program and explain its behavior. (Placing debugging information inside constructors and destructors is a very useful step in developing efficient and correct clases.)

```
#include <iostream>

class A
{
public:
        A(int n): xx(n)
        { cout << "A(int " << n << ") called" << endl; }

        A(double y): xx(y+0.5)
        { cout << "A(fl " << y << ") called" << endl; }

        ~A( )
        { cout << "~A( ) with A::xx = " << xx << endl; }

private:
        int xx;
};

int main( )
{
        cout << "enter main\n";
        int x = 14;
        float y = 17.3;
        A z(11), zz(11.5), zzz(0);

        cout << "\nOBJECT ALLOCATION LAYOUT\n";
        cout << "\nx is at " << &x;
        cout << "\ny is at " << &y;
        cout << "\nz is at " << &z;
        cout << "\nzz is at " << &zz;
        cout << "\nzzz is at " << &zzz;
        cout << "\n___________________________\n";
        zzz = A(x);
        zzz = A(y);
        cout << "exit main" << endl;
}
```

- **This program places a few objects and primitives on the stack (notice that if a double is passed**

0.5 is added to the value of the private data member xx).

- **Prints out the memory locations of each item placed on the stack.**
- **Then the program prints out the address in memeory where the stack variables are stored.**
- **The program then calles the A constructor with argument 14 (stack var x), but before it assigns 14 to xx it calls the desctructor to zzz which until the destructor was called held 0 in its xx private data member.**
- **A similar thing happens again where the A constructor is called with argument 17 (stack var y), but again before it assigns 17 to xx it calss the destructor to zzz which until the destructor was called held 14 in its xx private data memeber.**
- **The main program exits and the program then calls the destructor for the A class three times.**
  - **Once for the xx value of 17 (described on the previous line).**
  - **A seccond time for the xx value of 12 (12 beucae 11.5 was passed in and .5 was added).**
  - **And ~A() was called a final time for the xx value of 11 which was the first var to be placed on the stack.**

2. (6 points) Consider the class dynamicInt, which has single dynamic data member:

```
class dynamicInt
{
public:
        dynamicInt(int m = 0);

        ... // destructor

        dynamicInt(const dynamicInt& obj);    // copy constructor

        ... // assignment operator

        int getData() const;    // return value of dynamic member

private:
        int *ptr; // dynamic data member
};
```

(a) Define the destructor for dynamicInt.

```
~dynamicInt()
{
        delete ptr;
}
```

(b) Define the overloaded assignment opeartor.

```
dynamicInt& operator=(const dynamicInt& rhs)
{
        if (this != &rhs)
```

```
            {
                    delete ptr;
                    ptr= new int(*rhs.ptr);
            }
            return *this;
    }
```

(c) Implement the copy constructor.   Throw the memoryAllocationError
    exception if no dynamic memory is available.

```
        dynamicInt(const dynamicInt& obj)
      {
            ptr = new int(0);
            if (!ptr){
                    throw memoryAllocationError( "");
            }
            if (obj.ptr){
                    *ptr = *(obj.ptr);
          }
        }
```

(d) The function f() takes a dynamicInt object as an argument and has a
    dynamicInt object as its return value. The function is called in the main
    program.

```
        dynamicInt f(dynamicInt obj) // to make a copy copy consturctor called
        {
                dynamicInt localObj(5 + obj.getData());

                return localObj; ) // retruning object, copy constructor called

        }    // one call to destructor for obj and called another time for localObj

        void main( )
        {
                dynamicInt obj1(10), obj2;
                obj2 = f(obj1);
                cout << obj2.getData() << end;
        } destructor is called twice, one for obj1 and the other for obj2
```

(i) Indicate the number of times that the copy constructor is used.
    Comment to the right of the corresponding lines where the calls
    happen.

The constructor is called 2 times.

(ii) Indicate all of the times that the destructor is called.

The descturctor is called 4 times

(iii) What is the output value from obj2.getData()?

15

3. (2 points) Use the following declaration for parts (a)-(d):

```
rectangle *r;
```

where the class rectangle is defined below.

```
class rectangle
{
public:
        // constructor
          rectangle(double len = 0.0, double wid = 0.0)
                length(len), width(wid)
          {}

          // return the area
        double area() const
          { return length * width; }

          // return the perimeter
          double perimeter() const
          { return 2 * (length + width); }

          // change the dimensions of the rectange to len and wid
          void setSides(double len, double wid)
          {
             length = len;
             width = wid;
           }

          // return the length of the rectangle
        double getLength() const
          { return length; }

          // return the width of the rectangle
        double getWidth(void) const
          { return width; }

private:
        double length, width;
};
```

(a) Dynamically allocate a six-by-eight rectangle.

rectangle *r = new rectangle(6.0, 8.0);

(b) Use the dereference opeartor, *, and the dot operator to output the perimeter of the rectangle.

(*r).perimeter;

OUTPUT: 28

(c) Use the -> operator to output the area of the rectangle.

r->area();

OUTPUT: 48

(d) Using setSides() to, double the length and width of the rectangle.

r->setSides(2 * r->getLength(), 2 * r->getWidth());