OPEN BOOK, OPEN NOTES. NO Laptops or Cell Phones!

1. Constructors and destructors  **20 Points**

In the program `q1.cc` attached, identify where (what line number) each of the default constructors, int constructors, copy constructors, and destructors are called for each class `A` and `B`. Specify which line of code causes each of the above and a brief explanation of why the constructor was called. Be sure to note that the addition operator for classes `A` and `B` are both defined differently and implemented differently. As an example of how to fill in the table, one entry for the `A` int constructor is filled in.

<table>
<thead>
<tr>
<th>Class</th>
<th>Default Constructor</th>
<th>Line Number</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Default Constructor</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>int Constructor</td>
<td>Line Number</td>
<td>Explanation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>38</td>
<td>Declaration of local variable “a” with int argument</td>
</tr>
<tr>
<td>A</td>
<td>Copy Constructor</td>
<td>Line Number</td>
<td>Explanation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>Destructor</td>
<td>Line Number</td>
<td>Explanation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Default Constructor</td>
<td>Line Number</td>
<td>Explanation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>int Constructor</td>
<td>Line Number</td>
<td>Explanation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Copy Constructor</td>
<td>Line Number</td>
<td>Explanation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>Destructor</td>
<td>Line Number</td>
<td>Explanation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2. Inheritance and Virtual Functions  **20 Points**

What is printed by the program q2 attached? Explain your answer.

*Hint:* There should be 10 “Hello from” outputs.
3. Object Cloning  **20 Points**

What is printed by program q3.cc attached? Explain your answer. The printing is done by the `Hello` calls at lines 33 (which is in `Sub1` called from lines 42, 43, 44, and 45), plus the `Hello` call at line 47 (a total of 5 hello messages).
4. Memory Management  **20 Points** The code snippet in program q4.cc attached below has a fatal flaw and results in possible corruption of the heap and unexplained crashes. Explain what the flaw is and describe two possible solutions. One of your solutions should be efficient and avoid needless string copying. The other solution need not be concerned with efficiency. *You don’t need to provide code or pseudo-code for your solutions, just describe them.* You might not be familiar with the `strncpy` function at line 21. It simply says “copy 99 bytes from address s to address str”.

```cpp
// Sample code snippet
strncpy(s, str, 99);  // Line 21
```
5. Discrete Fourier Transform  **20 Points**

Suppose that we implemented Discrete Fourier Transform (DFT) using a naive approach, simply using the basic definition for the DFT given in equation (1) in the DFT assignment handout. We executed this implementation on a sample set consisting of 256 elements, and observed an execution time of 5 seconds. Since this execution time seems too long, we decided to implement a more efficient version of the algorithm, using the Danielson–Lanczos binary decomposition method and the Cooley–Tukey bit–reversal trick that we used for our FFT lab.

Estimate the running time for the new, efficient algorithm, when running on the same sample set of 256 elements. Assume that for both implementations we pre–computed the $W$ array (as we did in our implementation). Also, you can ignore any one–time overhead such as the time needed to read in the sample set from disk, to print out results, and to pre–compute the $W$ values. Explain your answer, state any assumptions you made, and show your work. Clearly, there is no exact “right answer”. Your answer must be reasonable, your assumptions must be valid, and your answer must be consistent with your assumptions.
// Code for ECE3090 midterm, QUESTION 1 - Constructors and Destructors

class A {
public:
    A(); // Default constructor
    A(int); // int Constructor
    A(const A&); // Copy constructor
    ~A(); // Destructor
    A operator+(const A& rhs) const; // Addition operator

public:
    int x; // Single data member
};

A A::operator+(const A& rhs) const
{
    A r(x + rhs.x);
    return r;
}

class B {
public:
    B(); // Default Constructor
    B(int); // int Constructor
    B(const B&); // Copy constructor
    ~B(); // Destructor
    B operator+(B rhs) const; // Addition operator

public:
    int x; // Single data member
};

B B::operator+(B rhs) const
{
    return B(x + rhs.x);
}

int main()
{
    A a(1);
    B b(2);
    a = a + a;
    b = b + b;
}

Program q1.cc
// Code for ECE3090 midterm, QUESTION 2 - Inheritance and Virtual Functions

class Base
{   // Define a base class
   public:
      void Sub1();
      virtual void Sub2() = 0;
      virtual void Sub3();
      virtual void Sub4();
};
class A : public Base
{   // Class A derives from Base
    public:
       void Sub1();
       void Sub2();
};
class B : public Base
{   // Class B derives from Base
    public:
       void Sub1();
       void Sub2();
       void Sub4();
};

// Base Class Methods
void Base::Sub1()
{
    cout << "Hello from Base::Sub1()" << endl;
}
void Base::Sub3()
{
    cout << "Hello from Base::Sub3()" << endl;
    Sub1(); // DON'T MISS THIS CALL IN YOUR ANSWER
    Sub4(); // DON'T MISS THIS CALL IN YOUR ANSWER
}
void Base::Sub4()
{
    cout << "Hello from Base::Sub4()" << endl;
}

// Class A Methods
void A::Sub1() { cout << "Hello from A:Sub1()" << endl; }
void A::Sub2() { cout << "Hello from A:Sub2()" << endl; }

// Class B Methods
void B::Sub1() { cout << "Hello from B:Sub1()" << endl; }
void B::Sub2() { cout << "Hello from B:Sub2()" << endl; }
void B::Sub4() { cout << "Hello from B:Sub4()" << endl; }

// A Helper Subroutine
void Sub(Base& x)
{
    x.Sub3();
    x.Sub2();
    x.Sub1();
}

int main()
{
    A a;
    B b;
    Sub(a);
    Sub(b);
}

Program q2.cc
// Code for ECE3090 midterm, QUESTION 3 - Object Cloning

class Base {
    public:
        virtual void Hello() { cout << "Hello from Base" << endl; }
        virtual Base* Clone() { return new Base(*this); }
    }

    // A derives from Base
    class A : public Base {
        public:
            virtual void Hello() { cout << "Hello from A" << endl; }
            virtual Base* Clone() { return new A(*this); }
    }

    // B derives from Base
    class B : public Base {
        public:
            virtual void Hello() { cout << "Hello from B" << endl; }
    }

    // C derives from B
    class C : public B {
        public:
            virtual void Hello() { cout << "Hello from C" << endl; }
            virtual Base* Clone() { return new C(*this); }
    }

    void Sub1(Base& p) {
        Base* newBase = p.Clone();
        newBase->Hello();
    }

    int main() {
        Base base;
        A a;
        B b;
        C c;
        Sub1(base);
        Sub1(a);
        Sub1(b);
        Sub1(c);
        Base d(a.Clone()); // Copy constructor
        d.Hello();
    }

    Program q3.cc
class A {
public:
A();    // Constructor
A(char*); // Constructor
~A();    // Destructor
public:
char* str;
};

A::A()
{
    str = new char[100];
    str[0] = '\0'; // String initially empty
}

A::A(char* s)
{
    str = new char[100];
    strncpy(str, s, 99); // Set initial string value
    str[99] = '\0';
}

A::~A()
{
    delete [] str; // Return the memory to heap
}

int main()
{
    A a1;
    A a2(a1);
    A a3 = a2;
    A a4;
    a4 = a1;
}

Program q4.cc