ECE3090 **Fall Semester, 2009**

Midterm Exam

Oct 12, 2009 *NAME:*_____

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 constuctors, 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.

A **Default Constructor**	**Line Number**	**Explanation**
A **int Constructor**	**Line Number**	**Explanation**
	38	Declaration of local variable "a" with int argument
A **Copy Constructor**	**Line Number**	**Explanation**
A **Destructor**	**Line Number**	**Explanation**
B **Default Constructor**	**Line Number**	**Explanation**
B **int Constructor**	**Line Number**	**Explanation**
B **Copy Constructor**	**Line Number**	**Explanation**
B **Destructor**	**Line Number**	**Explanation**

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`".

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.

```
1  // Code for ECE3090 midterm, QUESTION 1 - Constructors and Destructors
2
3  class A {
4  public:
5    A();         // Default constructor
6    A(int);      // int Constructor
7    A(const A&); // Copy constructor
8    ~A();        // Destructor
9    A operator+(const A& rhs) const; // Addition operator
10 public:
11   int x;       // Single data member
12 };
13
14 A A::operator+(const A& rhs) const
15 {
16   A r(x + rhs.x);
17   return r;
18 }
19
20 class B {
21 public:
22   B();         // Default Constructor
23   B(int);      // int Constructor
24   B(const B&); // Copy constructor
25   ~B();        // Destructor
26   B operator+(B rhs) const; // Addition operator
27 public:
28   int x;       // Single data member
29 };
30
31 B B::operator+(B rhs) const
32 {
33   return B(x + rhs.x);
34 }
35
36 int main()
37 {
38   A a(1);
39   B b(2);
40
41   a = a + a;
42   b = b + b;
43 }
```

Program q1.cc

```
1  // Code for ECE3090 midterm, QUESTION 2 - Inheritance and Virtual Functions
2
3  class Base
4  { // Define a base class
5  public:
6            void Sub1();
7    virtual void Sub2() = 0;
8    virtual void Sub3();
9    virtual void Sub4();
10 };
11 class A : public Base
12 { // Class A derives from Base
13 public:
14   void Sub1();
15   void Sub2();
16 };
17 class B : public Base
18 {// Class B derives from Base
19 public:
20   void Sub1();
21   void Sub2();
22   void Sub4();
23 };
24 // Base Class Methods
25 void Base::Sub1(){ cout << "Hello from Base::Sub1()" << endl;}
26 void Base::Sub3()
27 {
28   cout << "Hello from Base::Sub3()" << endl;
29   Sub1();  // DON'T MISS THIS CALL IN YOUR ANSWER
30   Sub4();  // DON'T MISS THIS CALL IN YOUR ANSWER
31 }
32 void Base::Sub4(){ cout << "Hello from Base::Sub4()" << endl;}
33
34 // Class A Methods
35 void A::Sub1() { cout << "Hello from A:Sub1()" << endl; }
36 void A::Sub2() { cout << "Hello from A:Sub2()" << endl; }
37 // Class B Methods
38 void B::Sub1() { cout << "Hello from B:Sub1()" << endl; }
39 void B::Sub2() { cout << "Hello from B:Sub2()" << endl; }
40 void B::Sub4() { cout << "Hello from B:Sub4()" << endl; }
41
42 // A Helper Subroutine
43 void  Sub(Base& x)
44 {
45   x.Sub3();
46   x.Sub2();
47   x.Sub1();
48 }
49
50 int main()
51 {
52   A a;
53   B b;
54   Sub(a);
55   Sub(b);
56 }
```

Program q2.cc

```
1  // Code for ECE3090 midterm, QUESTION 3 - Object Cloning
2
3  class Base {
4  public:
5    virtual void  Hello() { cout << "Hello from Base" << endl;}
6    virtual Base* Clone() { return new Base(*this);}
7  };
8
9  // A derives from Base
10 class A : public Base {
11 public:
12   virtual void  Hello() { cout << "Hello from A" << endl;}
13   virtual Base* Clone() { return new A(*this);}
14 };
15
16 // B derives from Base
17 class B : public Base {
18 public:
19   virtual void  Hello() { cout << "Hello from B" << endl;}
20 };
21
22 // C derives from B
23 class C : public B
24 {
25 public:
26   virtual void  Hello() { cout << "Hello from C" << endl;}
27   virtual Base* Clone() { return new C(*this);}
28 };
29
30 void Sub1(Base& p)
31 {
32   Base* newBase = p.Clone();
33   newBase->Hello();
34 }
35
36 int main()
37 {
38   Base base;
39   A a;
40   B b;
41   C c;
42   Sub1(base);
43   Sub1(a);
44   Sub1(b);
45   Sub1(c);
46   Base d(*a.Clone());  // Copy constructor
47   d.Hello();
48 }
```

Program q3.cc

```
1  // Code for ECE3090 midterm, QUESTION 4 - Memory Management
2
3  class A {
4  public:
5    A();     // Constructor
6    A(char*);// Constructor
7    ~A();    // Destructor
8  public:
9    char* str;
10 };
11
12 A::A()
13 {
14   str = new char[100];
15   str[0] = '\0'; // String initially empty
16 }
17
18 A::A(char* s)
19 {
20   str = new char[100];
21   strncpy(str, s, 99);  // Set initial string value
22   str[99] = '\0';
23 }
24
25 A::~A()
26 {
27   delete [] str; // Return the memory to heap
28 }
29
30 int main()
31 {
32   A a1;
33   A a2(a1);
34   A a3 = a2;
35   A a4;
36   a4 = a1;
37 }
```

Program q4.cc