CS 60
MID-TERM EXAM (TYPE B)
May 18, 2010
WRITE ALL YOUR ANSWERS ON SPACE PROVIDED.
ANSWER ALL QUESTIONS. TOTAL POINTS ARE 100.
ONLY STUDENTS THAT ARE REGISTERED FOR THIS COURSE MAY TAKE THIS EXAM.

NAME:_____

1 { Circle for each part True or False depending whether or not the statement is true or false. Each one is worth 1 point}

- { True or False } The function `new` returns a pointer to a block of memory all of which has been initialized to zero.
- { True or False } The memory allocated through `new` is in the area of memory called a Heap.
- { True or False } In C++ we are allowed to pass a `struct` to a function, and a function can return a `struct`.
- { True or False } When overloading `==` for a Class the two objects being compared must belong to the same class.
- { True or False } In C++ every Class must have a copy constructor and assignment operator (`operator=`) defined by the user.
- { True or False } The inline function `inline float square(float x) {return x*x;}` computes the square of the argument correctly when used in `float y = square(5.0);` but not when `float y = square(4.5 + 7.5);` is invoked because `*` has higher precedence than `+`.
- { True or False } To use `gdb` on the program `gdbfile.cpp` we should compile it with the command `g++ -d gdbfile.cpp`.
- { True or False } If we have a Class called Base (with several member variables), then the line `Base b(4);` invokes the copy constructor for that class.
- { True or False } If we have a Class called Base, then the line `Base f(b);` invokes the copy constructor for that class if `b` is an object of Class Base.
- { True or False } The destructor for the Class X is invoked only when an object for Class X was created dynamically (through `new`), but not when it is a local `auto` variable in a function.

- { True or False } If we have a Class called Base, then the line `Base *h = new Base;` invokes the assignment operator for Class Base.
- { True or False } If we have a Class called Base, executing the statement `Base *j = new Base[5];` and later on executing the line `delete[];`, causes the destructor for Class Base to be invoked five times.
- { True or False } When overloading `+` for the class `CL`, the operator `+` may be used for expressions of the form `a + b`, but not for expressions of the form `a + b + c`, where a, b, and `c` are objects of class `CL`.
- { True or False } In C++ every function needs to be a member function of at least one Class.
- { True or False } Class constructors can be `virtual`.
- { True or False } Class destructors can be `virtual`.

2 Quick Questions

(a) [3 points] How do you implement an #include guard? (I.e, what lines are added to the file?)

(b) [2 points] What will this C code segment print out:

```
int x = 0100;
printf("The answer is %d\n", x);
```

(c) [4 points] Name the four functions that are automatically created for you when you define a C++ class (if you don't define them yourself).

(d) [2 points] What is the value of rx after this code is executed:

```
int x = 50;
int &rx = x;
rx += 50;
```

(e) [4 points] What does the following code print?

```
#include <iostream>
using namespace std;
int main(void)
{
   int y=7;
   int i=3;
   int *pi=&i;
   int **ppi=&pi;
   pi = &y;
   cout << *pi << " " << **ppi << endl;
   pi = &i;
   cout << *pi << " " << **ppi << endl;
}
```

(f) [4 points] For the code given below clearly indicate what the `cout` command prints.

```
#include <iostream>
using namespace std;
int main(void)
 {int *p = new int[3];
  p[0] = 40; p[1] = 31; p[2] = 66;
  cout << p[0] << " " << p[1] << endl;
  p++;
  cout << p[0] << " " << p[1] << endl;
  p--;
  delete [] p;
 }
```

3 More Questions

(a) [2 points] Write C++ code in procedure `func` to assign to variable y in mygvar the value of parameter y (in func).

```
namespace mygvar {
  int x;
  int y;
}
void func(int y)
{ // Write code to assign to variable y in mygvar the value of
  // parameter y (in func).
}
```

(b) [6 points] What is the printed output when executing the following C++ code?

```
#include <iostream>
using namespace std;
class Base
{
 public:
  Base() { cout << "Base\n"; }
  virtual void bar() { cout << "Base::bar()\n"; }
  void foo() { cout << "Base::foo()\n"; }
};
class Derived : public Base
{
 public:
  Derived() { cout << "Derived\n"; }
  void bar() { cout << "Derived::bar()\n"; }
  void foo() { cout << "Derived::foo()\n"; }
};
int main(void)
{
  Derived d;
  Base b1 = d;
  Base *b2 = &d;
  b1.bar();
  b1.foo();
  b2->bar();
  b2->foo();
}
```

(c) [6 points] What does the following C++ code print in the standard output?

```
#include <iostream>
using namespace std;
 void XX( int &, int &, int);
 void YY( int &, int);
int main()
{int a, b, c;
 a = 1; b = 3; c = 2;
 XX(a,b,c);
 cout << "Main: a: " << a << " b: " << b << " c: " << c << endl;
 XX(c,b,a);
 cout << "Main: a: " << a << " b: " << b << " c: " << c << endl;
}

void XX(int & a, int &b, int c)
{
 YY(a,b);
 cout << "XX: a: " << a << " c: " << c << endl;
 YY(c,c);
 cout << "XX: c: " << c << " b: " << b << endl;
 return;
}

void YY(int & x, int y)
{
  x = x + y;
  y = 2 * y + 2 * x;
}
```

(d) [8 Points] What does the following program residing in three files print when it is compiled (`g++ qa.C qb.C qc.C`) and then it is executed. The three files are:
This is file qa.C

```
#include <iostream>
using namespace std;
int xcount= 0 ;

int main()
{ void xxx(int);
  int yyy(void);
  xcount = 35;
  cout <<"Main " << xcount << endl;
  xxx(3);
  cout <<"Main " <<xcount << endl;
  xxx(yyy());
  cout <<"Main " <<xcount << endl;
  yyy();
  cout <<"Main " <<xcount << endl;
  return 0;
}
```

This is file qb.C

```
#include <iostream>
using namespace std;
static int xcount = 0;

void xxx(int n)
{ cout << "Value to be displayed is " << n << endl;
  cout <<"xxx " << ++xcount << endl;
}
```

This is file qc.C

```
#include <iostream>
using namespace std;
extern int xcount;

int yyy(void)
{ cout <<"yyy " << ++xcount << endl;
   {static int xcount = 0;
    cout <<"yyy in " <<  ++xcount << endl;;
   }
  return 3;
}
```

(e) [4 points] What does the following code print in the standard output?

```
#include <iostream>
using namespace std;
int main()
{
int X(int n,int m=5); // <---
int x = X(6,8);
    x = X(X(8),X(5));
}
int X(int n, int m = 5)
{static int x = 0;
 x++;
 cout << x << " " << n << " " << m << endl;
 return n-m;
}
```

(f) [2 points] Now change the program so that the line with the "arrow comment" is changed to

```
int X(int n,int m=4); // <---
```

Does the program print exactly the same output? Why or why not?

(g) [5 Points] Consider the following code that uses class `L` defined below. What does the program print? Were there any memory leaks? Why or why not?

```
#include <iostream>
using namespace std;
class L
 {private: int i;
           int *j;
  public:
          L(int a, int b) { i=a; j = new int; *j = b;}
         ~L() {delete j;}
          void prints() { cout << i << " " << *j << endl;}
 };
int main()
 { L p(5,9);
   L * q = new L(3,6);
   p.prints();
   q->prints();
   p = *q;
   delete q;
   p.prints();
 }
```

4 C Code

(a) [10 points] Write a member function `Countit()` for the Class ListNode to count and return the number of nodes in the Singly-linked list that is linked through the pointer `next`. When this function is invoked initially pointer `this` will point to the first element in the list.

```
class ListNode {
private:
    ListNode  *next;
    int       val;
public: ...
};
```

(b) [22 Points] In class we discussed a procedure similar to the following one to add an element to an ordered list in such a way that the values stored in the list are sorted (from smallest to largest). In this case the list has a header node (which is a node type `struct list`. If the element to be inserted is in the list, then it will remain unchanged. Now modify the procedure so that when the element to be inserted is in the list instead of doing nothing, it deletes the element from the list and deletes the next element from the list (if it exists). All other cases are the same as before.

```
typedef struct list{
    int val;
    struct list* next;
  } List;
void addit(List *header, int w)
 { List *ptr,*temp;
   ptr=header;
   while(ptr->next != NULL)
    {if (ptr->next->val > w) break;
     if (ptr->next->val == w) return;
     ptr= ptr->next;
    }
   temp= (List *) calloc(1,sizeof(List));
   temp->next=ptr->next;
   temp->val=w;
   ptr->next=temp;
 }
```