Quiz Number 2 – Solutions
(Closed book/Closed Notes: 10 minutes)
Tuesday, April 04, 2006

Question 1: Consider the following piece of code. Write down the output in the space provided below (on next page) when the program in main.cc is executed.

```
// test.h
class CtorDtor
{
    private:
        int num;

    public:
        CtorDtor(int);
        ~CtorDtor();
};

// test.cc
#include <iostream>
#include "test.h"
using namespace std;

CtorDtor::CtorDtor(int a)
{
    this->num = a;
    cout << "In ctor" << a << endl;
}

CtorDtor::~CtorDtor()
{
    cout << "In dtor" << num << endl;
}

// main.cc
#include <iostream>
#include "test.h"
using namespace std;

CtorDtor a(1);

void create()
{
    CtorDtor b(2);
    static CtorDtor c(3);
    CtorDtor d(4);
}

int main()
{
    CtorDtor e(5);
    static CtorDtor f(6);
    create();
    CtorDtor g(7);
    return 0;
}
```
Write your answer below:

| In ctor1 | In ctor5 |
| In ctor6 | In ctor2 |
| In ctor3 | In ctor4 |
| In dtor4 | In dtor2 |
| In dtor7 | In dtor5 |
| In dtor3 | In dtor6 |
| In dtor1 |

**Question 2:** We know that compiler provides us with a default constructor. State as many reasons as possible due to which you will need to write a constructor of your own.

**Answer:**
- When you want to bring the initialization closer to the declaration
- When you have references as member variables in the class definition
- When you have `const` member variables in the class definition
- When your class encapsulates objects that do not have a default constructor

**Question 3:** State all the reasons why you would want to provide an initialization list?

**Answer:**
- When you have to (class definition contains `const`, references or objects that don’t have default constructors)
- For efficiency. The compiler initializes all the objects before the constructor executes. When an encapsulated object has a default constructor, the compiler calls the default constructor. Later in the constructor (or somewhere else), if you assign to that object, this sums up to two function calls for that object: one for default constructor by the compiler, and the other for assignment operation. On the other hand, if you put the initialization in the initialization list, only one function (the constructor) is called for that object.