# COP4530 – Data Structures, Algorithms and Generic Programming
## Recitation 5
## Date: 23rd September, 2011

**Lab topic:**

1) **Discus Quiz 3**
2) **Discussion on Iterators**
3) **Take Quiz 4**

## Iterators:

a) The concept in one line:

*"An iterator is an abstraction of the notion of a pointer to an element of a sequence "* -
B. Stroustrup (The C++ Programming Language, Chapter 19)

Examples of such sequences are arrays, vectors, singly-linked lists, doubly-linked lists, trees, input, and output. Each has its own appropriate kind of iterator.

b) Few Details:

- The iterator classes and functions are declared in namespace std and found in <iterator>.
- There is no concept of a ‘‘null iterator.’’ The test to determine whether an iterator points to an element or not is conventionally done by comparing it against the end of its sequence

.

- An iterator that points to an element is said to be valid and can be dereferenced (using *, [], or -> appropriately).

c) Example:

- An example of using an STL iterator: Traverse a list using iterator.

```
#include <iostream>

#include <list>

using namespace std;


typedef list<int> IntegerList;
```

```
int main()
{
    IntegerList    intList;

            //Insert elements
    for (int i = 1; i <= 10; ++i)
     intList.push_back(i * 2);


    //traverse elements
            for (IntegerList::const_iterator ci = intList.begin(); ci != intList.end(); ++ci)
     cout << *ci << " ";

    return 0;

}
```

- What changes are required in the following code so that the iterator iterates through every alternative element in the array.

```
#include <iostream>
#include <iterator>
using namespace std;

class myiterator : public iterator<input_iterator_tag, int>
{
  int* p;
public:
  myiterator(int* x) :p(x) {}
  myiterator(const myiterator& mit) : p(mit.p) {}
  myiterator& operator++() {++p;return *this;}
  myiterator operator++(int) {myiterator tmp(*this); operator++();          return tmp;}
  bool operator==(const myiterator& rhs) {return p==rhs.p;}
  bool operator!=(const myiterator& rhs) {return p!=rhs.p;}
  int& operator*() {return *p;}
};

int main () {
  int numbers[]={10,20,30,40,50};
  myiterator beginning(numbers);
  myiterator end(numbers+5);
  for (myiterator it=beginning; it!=end; it++)
        cout << *it << " ";
  cout << endl;
```

```
  return 0;
}
```

[Code Ref: http://www.cplusplus.com/reference/std/iterator/iterator/]