**Course Details**:

* **Course:** CSE-225 Lab (Data Structures and Algorithms)
* **Section:** 01
* **Time-slot:** ST 0940 AM : 1110 AM
* **Instructor:**

Md. Nazmul Hossain,

**email** : nazmul.hossain@northsouth.edu

**cell-phone no.** : 01731-492836

* **Facebook Group:**
  + **Name:** NSU-CSE-225L.1(Fall2017)
  + **Link:** https://www.facebook.com/groups/128896671092371/

**Pre-requisites**:

* CSE-115
* CSE-215

**Class and Course Policy**:

* Each lab class will carry attendance mark.
* Starting from the third lab class and onwards, there will be **graded practice in each class**.
* **Make-up policy**:
  + **Make-up exam due to medical reason**: You must take permission from the corresponding theory course faculty by writing an application for sitting for the makeup lab exam along with a set of copy of your valid medical documents.
  + **Make-up exam due to emergency/ personal/ family reasons:** You must take permission from the corresponding theory course faculty by writing an application (explaining the situation) for sitting for the makeup lab exam.
  + No make-up for ‘lab practice’
* **Tentative Percentage Breakdown:**
  + Attendance: **10%**
  + Lab-evaluation: **20%**
  + Midterm: **30%**
  + Lab Final Exam/Project: **40%**

**‘Academic Honesty’ policies**:

* Honest academic behavior will be of utmost importance.
* Any form of **dishonest academic behaviour** (copying of source codes, cheating during exams/ lab-evaluations) **will be very harshly dealt**.
* In both the cases of lab practices and lab exams, **the person copying** and **the person letting copy his/ her code**, will be **awarded zero as their lab practice/ exam score** during that class/ exam. Suspiciously similar code structure/ variable names/ solving techniques will be considered ‘copy’ works.

**How to write a class in C++**:

In C++, the following is the general format for a class declaration and definition:

**class class-name{**

private data variables and functions

access-modifiers:

respective data and functions

access-modifiers:

respective data and functions

**};**

**Here,** access-modifiers can be: public/ private/ protected (just like in JAVA). **By default**, functions and data declared within a C++ class are private to that class.

Suppose, **in JAVA**, you have written the following class named **DynamicArray-**

**public class DynamicArray{**

private int[] data;

public DynamicArray(int size)

{

data = new int[size];

}

public void insertItem(int index, int item)

{

data[index] = item;

}

public int getItem(int index)

{

return data[index];

}

**}**

Now, in the main method, you create an object of that above class like this:

public static void main(String[] args)

{

//create a dynamic array object with   
 //size = 10

DynamicArray d = new DynamicArray(10);

// calling the JAVA garbage collector to free the   
 // allocated memories

System.gc();

}

**------------------------------------------**

Now, if you convert the above JAVA class into a C++ class, it’ll consist of the following different parts:

* The first part is the ‘header’ file **(**with the file extension **.h)** which will contain only the declarations of all the class variables and class functions, no implementation here.

**dynamicarray.h**

**#ifndef DYNAMICARRAY\_H\_INCLUDED**

**#define DYNAMICARRAY\_H\_INCLUDED**

**class DynamicArray{**

private:

int\* data;

public:

DynamicArray(int);

~DynamicArray();

void insertItem(int, int);

int getItem(int);

**};**

**#endif**

* The second part is the cpp file **(**with the file extension **.cpp)** which will contain only the definitions of all the class variables and class functions ‘declared’ in the previous class header file. You **MUST** have to **include** the header file inside this cpp file.

**dynamicarray.cpp**

**#include “dynamicarray.h”**

DynamicArray::DynamicArray(int size)

{

data = new int[size];

}

void DynamicArray::insertItem(int index, int item)

{

data[index] = item;

}

int DynamicArray::getItem(int index)

{

return data[index];

}

DynamicArray::~DynamicArray()

{

delete[] data;

}

Now, in the main c++ file (also sometimes called the **driver file**) named **main.cpp**, you create and manipulate a DynamicArray class object as described below:

**main.cpp**

**#include “dynamicarray.cpp”**

**#include <iostream>**

using namespace std;

**int main()**

**{**

**// Prompting the user to enter the size of the array**

cout<<“Enter the size of the array: ”<<endl;

int size;

**// Taking the input from the user and assigning that value to the int variable named size**

cin>>size;

**// Creating the DynamicArray class object with the specified size**

DynamicArray d(size);

**// Taking 10 inputs from the user and saving them inside the DynamicArray object created   
 // above**

int temp;

for(int i=0;i<size;i++)

{

cout<< “Enter value to be inserted at index = ”<<i<<endl;

cin>>temp;

d.insertItem(i, temp);

}

**// Printing all the integer values saved in the DynamicArray class object**

cout<< “The values stored are: ”;

int temp2;

for(int i=0;i<size;i++)

{

temp2 = d.getItem(i);

cout<< “Index = ”<<i; cout<< “, Value = ”<<temp2<<endl;

}

return 0;

**}**

**Home Assignment (Submit handwritten hardcopy on the next class)**:

Write down in point form all the steps required for creating and adding the **header** and **cpp** files to an already created CodeBlocks project as demonstrated during the Lab-01 class to avoid the ‘precompiled header’ dilemma.

**Hint:** Remember how the **dynamicarray.h** and **dynamicarray.cpp** files were manually created as text files, then extensions were changed to **.h** and **.cpp** extensions and then how they were added to the project.