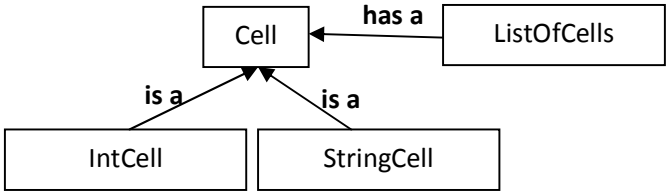
King Abdullah II School for Information Technology
Computer Science Department
Advanced Programming
Programming Homework #4: Pointer/Virtual/Overloading

Instructor: Dr. Jamal Alsakran **Due Date: Saturday 4/1/2014**

You are going to implement 4 classes (**Cell**, **IntCell**, **StringCell**, and **ListOfCells**) as shown in the figure below:



Class Cell:
An abstract class that has a pure virtual function Print. The implementation of the class is given below

```
#ifndef CELL_H
#define CELL_H

class Cell
{
public:
      Cell () {};
      virtual void Print () const = 0;
};
#endif
```

Class IntCell:
A class that publicly inherits class **Cell** and holds an integer value.

Data members:

```
      int value;
```

Member functions:

1. A constructor that sets **value**
2. A copy constructor
3. Assignment Operator Overloading (**operator=**)
4. **Print** function that displays **value**

Class StringCell: (Similar to IntCell except that it holds a string value)
A class that publicly inherits class **Cell** and holds a string value.

Data members:

```
      string value;
```

Member functions:

1. A constructor that sets **value**
2. A copy constructor
3. Assignment Operator Overloading **(operator=)**
4. **Print** function that displays **value**

Class ListOfCells:

A class that stores a list of Cells (combination of IntCell and StringCells)

```
#ifndef LISTOFCELLS_H
#define LISTOFCELLS_H

#include "Cell.h"

const int size = 10;

class ListOfCells
{
public:
      ListOfCells ();
      ListOfCells (const ListOfCells&);
      ~ListOfCells ();
      const ListOfCells& operator= (const ListOfCells&);
      void AddCell (Cell*);
      void Print () const;
private:
      int length;       // holds the current number of cells in the list
      Cell* list[size]; // list of pointers to Cell (abstract class)
};
#endif
```

Member functions:

1. A constructor that initializes **length** to 0, and allocates **list**
2. A copy constructor
3. A destructor that deallocates **list**
4. Assignment Operator Overloading **(operator=)**
5. **AddCell**: adds a cell to the end of the list and increases **length** by 1. **Note: if the list is full (length == size) this operation cannot proceed**
6. **Print:** displays all the values in the list. **Note: this function calls Print of IntCell/StringCell**

Main.cpp:

Have a main function to test the functionalities of your classes

Submission:

Upon the successful completion of the above tasks you should have the following files:
main.cpp, Cell.h, IntCell.h, StringCell.cpp, ListOfCells.h, IntCell.cpp, StringCell.cpp, and ListOfCells.cpp

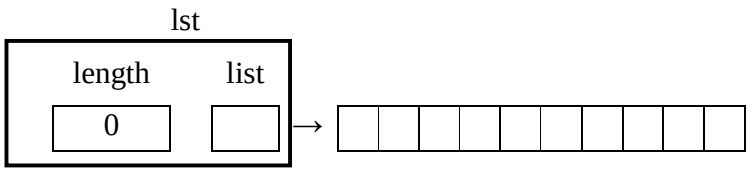
How to Submit:

1. Select all your files, right click and compress them to generate **StudentID_LastName_Section.zip.**
2. Go to **ELearning** and submit your homework

Here's a sample main function with some illustration:

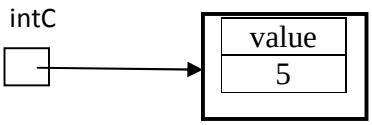
```
#include "IntCell.h"
#include "StringCell.h"
#include "ListOfCells.h"


void main ()
{
	// create a list of cells of size 10.
	// LisyOfCells constructor will set size to 10, length to 0, and allocate list of
	// 10 elements each of them is a pointer to list.
	ListOfCells lst;
```

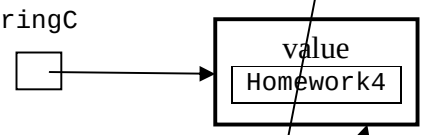



```
	// lst is now empty, so calling lst.Print () will display nothing.
	lst.Print ();

	// create an IntCell and pass 5 to value.
	IntCell* intC = new IntCell (5);
```

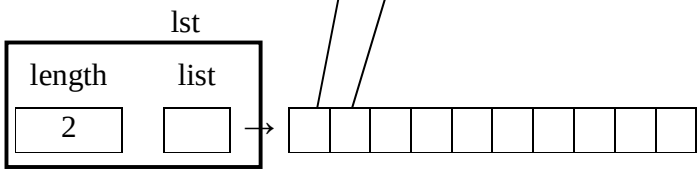



```
	// create a StringCell and pass 3.2 to value.
	StringCell* stringC = new StringCell ("Homework4");
```




```
	// add intC to lst. NOTE: length will be increased by 1.
	lst.AddCell (intC);

	// add stringC to lst. NOTE: length will be increased by 1.
	lst.AddCell (stringC);
```




```
	// lst has two elements now, so calling lst.Print () will display 5 and 3.2.
	lst.Print ();
```

```
        // make a copy of stringC by call copy constrcutor.
        StringCell s (*stringC);
        s.Print ();

        // make a copy of intC using assignment operator overloading.
        IntCell i (10);
        i = *intC;
        i.Print ();

        // delete doubelC and intC;
        delete stringC;
        delete intC;

        // ..... test more functionalities

}
```