Data Structures and Algorithms Lab

Instructions

Work on this lab individually. *Write **main** function first and keep on testing the functionality of each function once created.*

Program the following tasks in your **C++** compiler and then compile and execute them.

Email your solution (**.cpp**) file only to the following respective recipient till **Friday, March 19, 2021**.

DO NOT compress/zip your solution.

The email must be sent from your **official PUCIT email id**, otherwise it will **NOT BE ACCEPTED** and will be marked **ZERO**.

2 MARKS will be **DEDUCTED** if submission instructions are not followed.

The subject of the email should be the name of the lab e.g. **Lab 03**.

Degree	Recipient Email	Subject of Email
BSIT Morning	dsaubt01@gmail.com	Lab 03
BSIT Afternoon	dsaubt02@gmail.com	

You are strictly not allowed to add any other data-member/constructor/function in the class. You are also not allowed to change the name or prototype of any data-member/constructor/function.

ADT: PointList

Given the **Point** structure

```
struct Point
{
   float x, y;
};
```

Develop a class to store data items in an unordered list (**PointList**). Each data item in a point list is of type **Point**.

There will be four data members (private) of this class:

- A pointer to a **Point** structure that holds an **array of points** allocated dynamically according to the specified **maximum size.**
- An integer maxSize to store the maximum number of elements in the list.
- An integer curSize to hold the current number of the elements in the list.
- An integer cursor to hold the current position (index). If the list is empty it should hold -1.

Your class should support the following operations:

A. Constructor for creating a **List** of any size. The size will be specified through argument.

B. Destructor to free any memory resources occupied by an object.

C. `bool isEmpty(void)` returns *true* if a list is empty. Otherwise, returns *false*.

D. `bool isFull(void)` returns *true* if a list is full. Otherwise, returns *false*.

E. `void insert(Point newPoint)` inserts newPoint into the list. If the list is not empty, then inserts newPoint at the end. Otherwise, inserts newPoint as the first (and only) data item in the list. In either case, moves the cursor to newPoint.

F. `void showStructure(void)` outputs the data items in a list. If the list is empty, outputs "Empty list".

G. `Point getCursor(void)` returns a copy of the data item marked by the cursor.

H. `void gotoBeginning(void)` if a list is not empty, then moves the cursor to the data item at the beginning of the list.

I. `void gotoEnd(void)` if a list is not empty, then moves the cursor to the data item at the end of the list.

J. `bool gotoNext(void)` if the cursor is not at the end of a list, then moves the cursor to the next data item in the list and returns *true*. Otherwise, returns *false*

K. `bool gotoPrior(void)` if the cursor is not at the beginning of a list, then moves the cursor to the preceding data item in the list and returns *true*. Otherwise, returns *false*.

L. `void clear(void)` make the list empty.

M. `void replace(Point newPoint)` replaces the data item marked by the cursor with newPoint. The cursor remains at newPoint.

N. `void remove(void)` removes the data item marked by the cursor from a list. If the resulting list is not empty, the cursor should now be marking the data item that followed the deleted data item. If the deleted data item was at the end of the list, then moves the cursor to the data item at the beginning of the list.

No submission will be accepted after the DUE DATE.

B E S T O F L U C K