
1 Class Description

Design and implement a linked list class called `PointList`. The list will be composed of list nodes, objects of a `PointListNode` class, each of which contain an object of the `ExtVector` class and a pointer to the `next` list node.

1.1 Constructors

Implement (at least) the following constructors. Other constructors may be necessary.

For `PointList`:

- Null constructor - Empty `PointList`
- Copy constructor - Makes a deep copy of a `PointList`. **Must be recursive.**

For `PointListNode`:

- Null constructor - Empty `PointListNode`
- One parameter constructor - Parameter is a constant reference to an `ExtVector`
- Copy constructor - Makes a deep copy of a `PointListNode`. **Must be recursive.**

1.2 Destructor

Properly delete both `PointList` and `PointListNode` objects.

1.3 PointList Operations

Implement the `PointList` operations and **all** necessary `PointListNode` operations to support the listed operations:

- `insert(const ExtVector& e)` - Insert `ExtVector e` as the first node of the `PointList`.
- `append(const ExtVector& e)` - Append `ExtVector e` as the last node of the `PointList`.
- `remove(const ExtVector& e)` - Remove **all** occurrences of `PointListNodes` containing the same elements as `ExtVector e`.
- `count(const ExtVector& e)` - Return a count of the number of `PointListNodes` containing the same elements as `ExtVector e`.

1.4 PointList Size

The member function `size()` must return the number of elements (`PointListNodes`) in the `PointList`.

1.5 Assignment Operator

Make a deep copy of the RHS `PointList` and assign it to the LHS `PointList`. **Must be recursive.**

1.6 + Operator

Addition of the `PointLists` is simply concatenation of the `PointList` specified as the parameter with the `PointList` invoking the operator. Return the concatenated `PointList`.

1.7 Output

The insertion operator `<<` should print out the elements of the `PointList`. Print the contents of one `PointListNode` per line. Include the address contained in the `next` member. For example, if a list contained the vectors `{1.5, 2.3}` and `{3.4, 4.7, 5.5}` the output would be

```
<[1.5, 2.3] (0x21de8)>
<[3.4, 4.7, 5.5] (0)>
```

2 Programming Points

You **must** adhere to all of the following points to receive credit for this program.

1. Turn-in (print-outs and electronically) the files for this program.
2. You must submit all the files necessary to compile and link an executable program that utilized the `PointList` and `PointListNode` classes. This includes (but is not limited to) the following files (use the names listed below).
 - `pointlist.h` Contains the `PointList` class definition.
 - `pointlist.cpp` Contains the `PointList` member definitions.
 - `pointlistnode.h` Contains the `PointListNode` class definition.
 - `pointlistnode.cpp` Contains the `PointListNode` member definitions.
 - `driver.cpp` A *driver* program that tests the `PointList` class.
 - `extvec.h`, `extvec.cpp`, `extvec.cpp`, `vector.h`, `vector1.cpp`, `vector2.cpp`
 - `makefile` A makefile to compile the driver program. Note, the makefile must also compile all necessary files, be commented, and have a `make clean` option.
3. All `ExtVectors`, `PointListNodes`, and `PointLists` must be dynamically allocated with **no wasted space!** Be certain **no** memory leaks occur.
4. Perform appropriate error checking.