CS 60
Programming Assignment #8
Due Date: June 1, 2010 (5:00 pm) 80 Points.
Remember: Points will be deducted if turned in after the Due Date.
Deadline: June 2, 2010 (5:00 pm)
Remember: No homework will be accepted after the Deadline
You must work on this assignment independently.

For this assignment you will write a C++ program to manipulate a sorted singly-linked list similar in nature to the program in http://www.cs.ucsb.edu/~teo/cs60.s10/prog/l2.C The `l2.C` program just creates singly-linked sorted lists and it is organized in one file. The program you will write for this assignment should be stored in different files. The main difference between the code you will write and the one in `l2.C` is that instead of having a sorted linked list of objects of the same type, now you will have a sorted linked list of nodes that are objects of Class ListNode, DListNode (Derived) and SDListNode (Second Derived). Below you will find declarations for these classes and you must add appropriate member functions to them. At least a couple of the member functions in the ListNode Class must be `virtual` (besides function `whoami()`).

```
#ifndef LISTNODE_H
#define LISTNODE_H

#include <iostream>

// Definition of Class ListNode
class ListNode {
protected:  // Each object contains two values
    int data;
    ListNode *link;
public:   // These are the member functions for the Class ListNode
    virtual int whoami() {return 1;}

};
#endif //LISTNODE_H

#ifndef DLISTNODE_H
#define DLISTNODE_H

#include <iostream>

// Definition of Class DListNode
class DListNode : public ListNode {
protected:  // Each object contains two values
    int ddata;
public:   // These are the member functions for the Class DListNode
```

```
    int whoami() {return 2;}

};
#endif //DLISTNODE_H

#ifndef SDLISTNODE_H
#define SDLISTNODE_H

#include <iostream>

// Definition of Class SDListNode
class SDListNode  : public DListNode{
protected:  // Each object contains two values
    int sddata;
public:   // These are the member functions for the Class SDListNode
    int whoami() {return 3;}

};
#endif //SDLISTNODE_H
```

The Class List (similar to the one in `l2.C`) will contain as data one pointer to ListNode (`ListNode *first;`). The list will never contain two ListNodes with the same `data`; or two DListNodes with the same `data` and `ddata`; or two SDListNodes with the same `data`, `ddata` and `sddata`. The list will be sorted in ascending order of the `data` attribute of each object. There will be at most one ListNode, but there may be several DListNodes and SDListNodes, with the same `data` attribute. In this case we put the ListNode before all the DListNodes and SDListNodes (with the same `data` value). These DListNodes and SDListNodes (with same `data` value) are sorted in ascending order of the `ddata` attribute. There may be at most one DListNode, but there may be several SDListNodes, with the same `data` and `ddata` values. In this case we put the DListNode (with the same `data` and `ddata`) before all the SDListNodes. These SDListNodes, with same `data` and `ddata` values, are sorted in ascending order of the `sddata` attribute. But there may be at most one DListNodes with the same `data`, `ddata` `sddata` attributes. The example we give below illustrates the ordering (left-to-right and then top-down)

```
(4), (4 5), (4 7), (4 7 2), (4 7 5), (4 8), (5 1), (5 3 4), (5 4 3), ...,
(6), (8 3 4), (8 4), (9 6), (9 7), (10)
```

The above Class declarations, which are also given in the `map.html` web page, will help you get started. You may use any part(s) of that code in your homework.

Your main program will read in a set of commands to manipulate the lists with the different types of objects. In the map.html web page you will find a skeleton code that shows how to read in the data (you may use this code, but note that you need to add a few things to make it work). You will also find in the map.html page a sample input and output file. Your output format should mimic the sample output file. You should write a

constructor that will initialize a list to empty. In this assignment you will manipulate 9 different lists (named 1, 2 ... 9). The command lines are given below. You must print every command line just after you read it. By `i1, i2,` and `i3` below we mean that you should expect a positive integer for each of them. By `j` and `k` we mean DIFFERENT integers in the range 1, 2, ..., 9. Initially each of these lists is empty. i.e., the nine lists have zero elements.

- `insert j base i1`

  You will add a ListNode object with `data` value `i1` to the list `j`. If the list `j` already contains a ListNode object with `data` value `i1`, then the operation will do nothing. The list should be sorted as specified above.

- `insert j der i1 i2`

  You will add a DListNode object with `data` value `i1` and `ddata` value `i2` to the list `j`. If the list `j` already contains a DListNode object with `data` value `i1` and `ddata` value `i2`, then the operation will do nothing. The list should be sorted as specified above.

- `insert j sder i1 i2 i3`

  You will add an SDListNode object with `data` value `i1`, `ddata` value `i2`, and `sddata` value `i3` to the list `j`. If the list `j` already contains a SDListNode object with `data` value `i1`, `ddata` value `i2`, and `sddata` value `i3` then the operation will do nothing. The list should be sorted as specified above.

- `delete j base i1`

  You will delete the ListNode object with `data` value `i1` from the list `j` if it exists. If it is not in the list `j` the operation will do nothing.

- `delete j der i1 i2`

  You will delete the DListNode object with `data` value `i1` and `ddata` value `i2` from the list `j` if it exists. If it is not in the list `j` the operation will do nothing.

- `delete j sder i1 i2 i3`

  You will delete the SDListNode object with `data` value `i1`, `ddata` value `i2`, and an `sddata` value `i3` from the list `j` if it exists. If it is not in the list `j` the operation will do nothing.

- `member j base i1`

  You will print `true` if the ListNode object with `data` value `i1` is in the list `j`, otherwise print `false`.

- `member j der i1 i2`

  You will print `true` if the DListNode object with `data` value `i1` and `ddata` value `i2` is in the list `j`, otherwise print `false`.

- `member j sder i1 i2 i3`

  You will print `true` if the SDListNode object with `data` value `i1`, `ddata` value `i2`, and an `sddata` value `i3` is in the list `j`, otherwise print `false`.

- `compute j i1`

  This command computes and prints the total value of the list `j`. The total value of the list `j` is the sum of the values of the nodes in the list `j`. The value of a node of the list `j` is computed as follows. For a ListNode its value is `i1` times its `data` value. For an DListNode its value is `(i1+5)` times its `data` value plus `i1` times its `ddata` value. For an DListNode its value is `(i1+7)` times its `data` value plus `(i1+2)` times its `ddata` value plus `(i1+4)` times its `sddata` value.

- `print j`

  This will print the whole list `j` in the order it appears. For each object in the list `j` it prints the Class of object it is and then all its values (i.e., the `data` value for a ListNode; the `data` value and `ddata` value for a DListNode; and the `data` value, the `ddata` value, and `sddata` value for an SDListNode.

- `join j k`

  Move all the elements in list `k` to list `j`. Note that list `k` will be empty after this operation. Note that after the operation list `j` should not have repeated elements.

- `quit`

  Your program must delete all the objects your program has created and end.

```
You may assume that each input line is of the form indicated above.
You may also assume that there is a \n after the last character in each line, and
there are no other blanks (or symbols) in the input.  See the sample
input file in the class web page.
```

You must have a different function for each of the commands and you must use several `virtual` functions.

You must have a `.H` file and a `.C` files for each class, plus there must be a `main` program. You must use a `makefile`.

Turnin electronically to hw08@cs60. You must include all your .H, .C, makefile and and the `student.id` file.

There is an example in the map.html web page.