CS 60
Programming Assignment #6
Due Date: May 13, 2010 (11:59 PM). 70 Points.
Remember: Points will be deducted for homework turned in after the Due Date.
Deadline: May 14, 2010 (11:59 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 (variation of mini Google) multilinked structure. The problem is to keep a set of web pages each of which contains a set of index terms. To facilitate the assignment each web page will be represented by a positive integer (`int`). Also each index term will be a positive integer (`int`). Each web page contains exactly three index terms.

Each web page will be represented by an instance of `class WebPage`. The web pages will be in increasing order (with respect to the web page number) in a singly-linked list using the pointer `nextwp` in `class WebPage`.

```
class WebPage {
  private:
    int       wpnumber;   // Web page id
    WebPage*  nextwp;     // Pointer to web page
    ItLinks   itl[3];     //Index terms Links
  public:
  }


class ItLinks {
  private:
    int       IndexTerm;  // Web page id
    WebPage*  wplink;     // Pointer to web page
  public:
  }
```

The `class WebList` is used for the code for basic operations on the web pages. It is defined as follows:

```
class WebList {
  private:
    WebPage*  first;     // Pointer to first web page in list
    int       number;    // Total number of web pages in list
  public:
  }
```

Each index term will be represented by an instance of `class IndexTerm`. We will also maintain this list in increasing order (with respect to the index term number) in a singly-linked list using the pointer `nextit` in `class IndexTerm`.

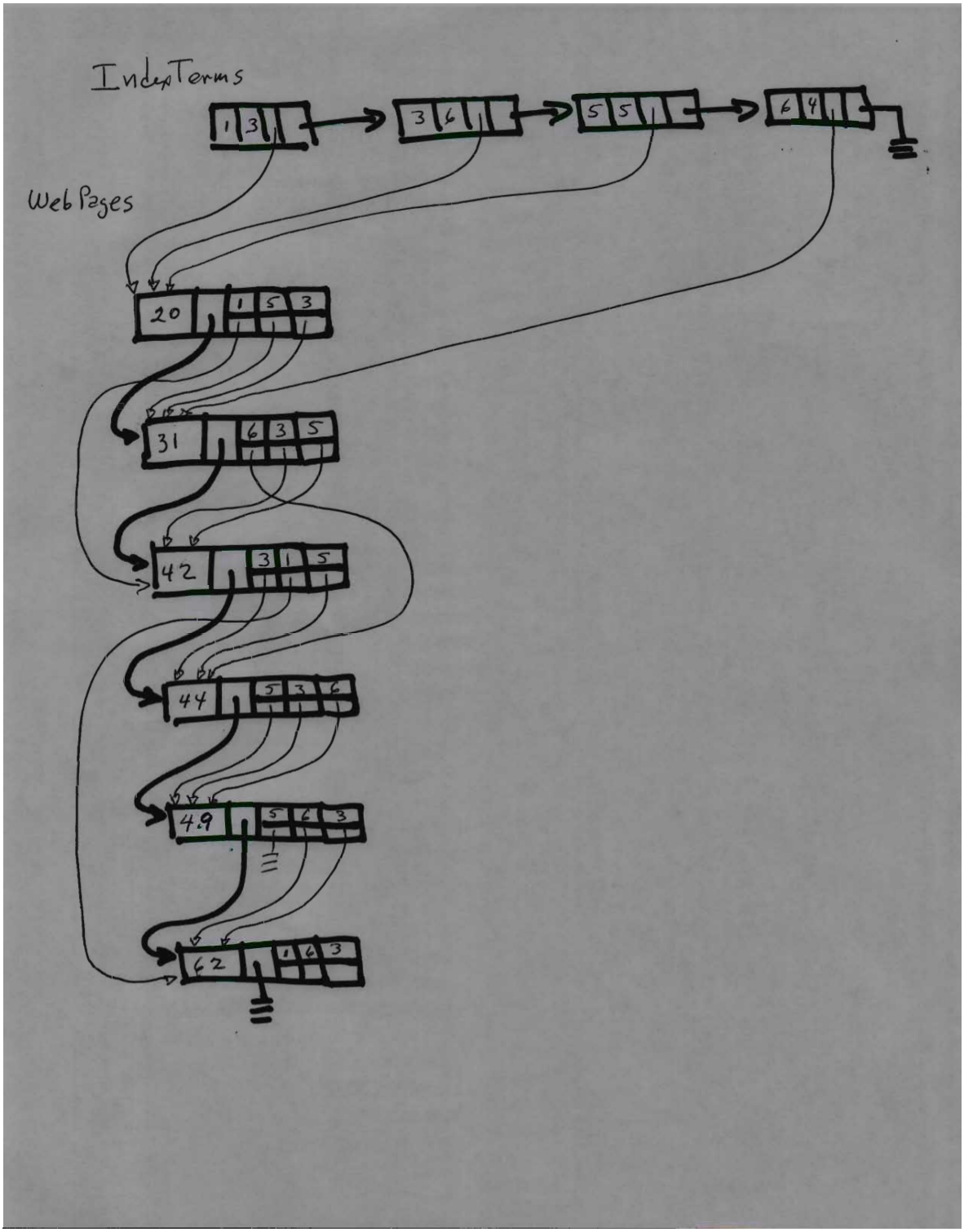
```
class IndexTerm {
  private:
    int        itnumber;       //Index term id
    int        numberoftimes;  //# of web pages that include itnumber
    WebPage*   link2webpage;   //Pointer to web page
    IndexTerm* nextit;          //Pointer to index term
  public:
  }
```

The `class ITList` is used for the code for basic operations on the web pages. It is defined as follows:

```
class ITList {
  private:
    IndexTerm*  first;   // Pointer to index term in list
    int       number;    // Total number of (different) index terms
  public:
  }
```

In addition there will be one sorted (with respect to page number) singly linked list for each index term. Consider index term 150. The singly linked list for index term 150 starts at the instance of the `class IndexTerm` that represents index term 150 and it uses the pointer `link2webpage`. This takes us to a web page (lets say it is web page 200) that uses index term 150. In the instance of `class WebPage` for web page 200, one of the entries in the `itl` array will have the `IndexTerm` equal to 150. Then in the corresponding entry `wplink` will either be Null (which means the list ends there) or it will point to an instance of a `class WebPage` that represents another web page. This other web page must have a number greater than 150. The remaining web pages that use the index term 150 are found by continuing the same search process just described. As you traverse the list, the web page numbers MUST appear in increasing order.

Below you will find an instance of our data structure.


IndexTerms
1 3
3 6
5 5
6 4
Web Pages
20
1 5 3
31
6 3 5
42
3 1 5
44
5 3 6
49
5 6 3
62
1 6 3

Your main program will read in a set of commands to manipulate the set of web pages. 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 map.html page a sample input and output file. **YOUR PROGRAM MUST MIMIC THE OUTPUT LINES IN THE SAMPLE OUTPUT FILE. YOU WILL LOSE POINTS IF YOU DO NOT FOLLOW THE PRINTING PATTERN IN THE SAMPLE OUTPUT FILE.** The command lines are given below. Note that there are NO `x` and `y` commands. You must print every command line just after you read it. By `m, i1, i2` or `i3` below we mean that you should expect a positive integer.

- `i m i1 i2 i3`

  You may assume that `i1, i2` and `i3` are all different. There is no ordering between the values of `i1, i2` and `i3`. This command indicates that there is a new web page `m` with three index terms denoted by `i1, i2` and `i3`. If the web page already exists, then the operation will do nothing. Otherwise a new instance of `class WebPage` must be created and it should be added to three lists (the list of web pages and the three lists of index terms). Note that if there was not a list for any of the index terms, they should be created at this point too.

- `d m`

  This indicates that you must delete the web page `m`. If `m` does not exist, then your program should not make any changes and nothing should be printed. When the page exists, the instance of `class WebPage` representing web page `m` should be deleted from the list of web pages and from the three index terms used by the web page should be deleted from the three individual lists. If any of these individual lists become empty (i.e., the index term(s) is not used in any remaining web page), they should also be deleted. I.e., thay should be deleted from the List of Index Terms. When one deletes or removes an object from the data structure, one must also free the space it uses.

- `t m`

  This indicates that you must print *true* if you already have the web page `m` in the system, and *false* otherwise.

- `c i1`

  If the index term `i1` does not exist, then nothing should be printed. Otherwise, you must print the number of web pages that include the index term `i1`. Note that there must be at least one such web page if the index term is represented by your program.

- `p i1`

  Same is `c i1`, but instead of printing the number of web pages, you should print the id of each web page that contains index term `i1`. **Note that in the printout the web pages must be in INCREASING ORDER.**

- `l`

  Print the number of web pages currently in the system.

- `r`

  Print the id of all the web pages (in increasing order) currently in the system.

- `n`

  Print the number of (different) index terms currently in the system.

- `s`

  Print the id of all the (different) index terms (in increasing order) currently in the system.

- `q`

  Your program must delete all the elements from all the lists and end. You must free all the space that was allocated dynamically.

```
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.
```

The different classes must have member functions appropriate for the class. We want to see `C++-like` code, rather than `C-like` code. In other words, you must have "approriate" member functions for the different classes.

You must have a `.H` file and a `.C` file for each of your classes and one file for `main`. You must use a `makefile`. The executable produced when we type `make` should be the file called `executeit`.

Turnin electronically to hw06@cs60. You must include all your .H, .C, makefile and the student.id files.

Example: The resulting state after of the list after each command is given below. We also indicate what the command will output (if nothing is printed then it is left blank). Note that the "state of the system" should NOT be printed. It is included below to illustrate the information your program should be maintaining. The initial condition is that the web page list and the list of index terms must be empty.

```
INPUT COMMAND    Resulting State
-----------      ----------------------
                    State:
                     Web Pages
                     Index terms
i 40 5 9 8       Output: i 40 5 9 8
                    State:
                     Web Pages
                      40:  5  9  8
                     Index terms
                      5 8 9
i 30 3 9 8       Output: i 30 3 9 8
                    State:
                     Web Pages
                      30:  3  9  8
                      40:  5  9  8
                     Index terms
                      3  5  8  9
i 52 4 7 6       Output: i 52 4 7 6
                    State:
                     Web Pages
                      30:  3  9  8
                      40:  5  9  8
                      52:  4  7  6
                     Index terms
                      3  4  5  6  7  8  9
i 55 3 9 5       Output: i 55 3 9 5
                    State:
                     Web Pages
                      30:  3  9  8
                      40:  5  9  8
                      52:  4  7  6
                      55:  3  9  5
                     Index terms
                      3  4  5  6  7  8  9
i 30 1 12 9        Output: i 30 1 12 9
                    State:
```

```
                    Web Pages
                     30:  3  9  8
                     40:  5  9  8
                     52:  4  7  6
                     55:  3  9  5
                    Index terms
                     3  4  5  6  7  8  9
d 52             Output: d 52
                   State:
                    Web Pages
                     30:  3  9  8
                     40:  5  9  8
                     55:  3  9  5
                    Index terms
                     3  5  8  9
i 52 3 4 7       Output: i 52 3 4 7
                   State:
                    Web Pages
                     30:  3  9  8
                     40:  5  9  8
                     52:  3  4  7
                     55:  3  9  5
                    Index terms
                     3  4  5  7  8  9
t 30             Output: t 30
                         true
                   State:
                    Web Pages
                     30:  3  9  8
                     40:  5  9  8
                     52:  3  4  7
                     55:  3  9  5
                    Index terms
                     3  4  5  7  8  9
t 45             Output: t 45
                         false
                   State:
                    Web Pages
                     30:  3  9  8
                     40:  5  9  8
                     52:  3  4  7
                     55:  3  9  5
                    Index terms
                     3  4  5  7  8  9
c 9               Output: c 9
                          3
                   State:
                    Web Pages
                     30:  3  9  8
                     40:  5  9  8
                     52:  3  4  7
                     55:  3  9  5
                    Index terms
                     3  4  5  7  8  9
```

```
c 30                Output: c 30
                     State:
                      Web Pages
                       30:  3  9  8
                       40:  5  9  8
                       52:  3  4  7
                       55:  3  9  5
                      Index terms
                       3  4  5  7  8  9
p 3                  Output: p 3
                             30
                             52
                             55
                     State:
                      Web Pages
                       30:  3  9  8
                       40:  5  9  8
                       52:  3  4  7
                       55:  3  9  5
                      Index terms
                       3  4  5  7  8  9
p 30                 Output: p 30
                     State:
                      Web Pages
                       30:  3  9  8
                       40:  5  9  8
                       52:  3  4  7
                       55:  3  9  5
                      Index terms
                       3  4  5  7  8  9
i 22 2 3 7           Output: i 22 2 3 7
                     State:
                      Web Pages
                       22:  2  3  7
                       30:  3  9  8
                       40:  5  9  8
                       52:  3  4  7
                       55:  3  9  5
                      Index terms
                       2  3  4  5  7  8  9
l                    Output: l
                             5
                     State:
                      Web Pages
                       22:  2  3  7
                       30:  3  9  8
                       40:  5  9  8
                       52:  3  4  7
                       55:  3  9  5
                      Index terms
                       2  3  4  5  7  8  9
r                    Output: r
                             22
                             30
```

```
                        40
                        52
                        55
                    State:
                     Web Pages
                      22:  2  3  7
                      30:  3  9  8
                      40:  5  9  8
                      52:  3  4  7
                      55:  3  9  5
                     Index terms
                      2  3  4  5  7  8  9
n                   Output: n
                        7
                    State:
                     Web Pages
                      22:  2  3  7
                      30:  3  9  8
                      40:  5  9  8
                      52:  3  4  7
                      55:  3  9  5
                     Index terms
                      2  3  4  5  7  8  9
s                   Output: s
                        2
                        3
                        4
                        5
                        7
                        8
                        9
                    State:
                     Web Pages
                      22:  2  3  7
                      30:  3  9  8
                      40:  5  9  8
                      52:  3  4  7
                      55:  3  9  5
                     Index terms
                      2  3  4  5  7  8  9
q                   Output: q
                    State:
                     Web Pages
                     Index terms
```