ECS 40 **Program #8 (50 points, 1.5 hours)** **Winter 2016**

Due : Monday, March 14th, at 11:59 P.M. in p8 of cs40a

New concepts: multimap, map, STL algorithms. Executable must be p8.out

Files to submit: indexpager.cpp, indexpager.h, Makefile, and authors.csv.

For this assignment you will write an IndexPager class that will create an index of all the words in a file. The index will list the line numbers that each word is on. The words are sorted alphabetically, and the line numbers are listed in ascending order. If a word appears in consecutive lines then the index should use a dash between the first line number and the last consecutive line number, else the line numbers should be simply separated by commas. See the demonstration on the back of this sheet.

The file name is passed as the first command line parameter. If the second command line parameter is a 1, then the Pager class will read the file and display it. If the second parameter is not a 1 then your IndexPager class must read the file and display its indices. Your IndexPager class must be derived from Pager class. The ProcessKey() method of the Pager class scrolls up or down based on whether a 'b' or 'f' is entered by the user. The program ends when 'q' is entered by the user. You will need provide at least a read() method for IndexPager.

You may NOT use any char* or arrays in this assignment. You will find the IntToString function of Pager useful. You must use a map to provide a mapping between each word and a unique integer, called WordKey here, assigned to that word. You must use a multimap to provide a mapping between WordKey and each of the line numbers the associated word appears on. After you've created these mappings from the file, you can create strings for the Text. You can store the line numbers associated with each word in a vector of ints and then use the sort algorithm. Once the line numbers are sorted you can work through the vector to create the appropriate string for each word.

A "word" is any set of consecutive alphabetic characters. You need not worry about errors on the command line or keyboard entry. I have provided pager.cpp, pager.h, and main.cpp in ~ssdavis/40/p8. You may not alter any of the files I have provided!

```
[ssdavis@lect1 p8]$ p8.out pager.cpp 2
IntToString 6, 13.
Pager 6, 15, 34, 43.
ProcessKey 15, 32.
Text 20-21, 39, 47-48, 51.
TopLine 19-21, 27-29, 45, 50.
WritePage 22, 30, 43, 52.
b 15, 25, 31.
back 15, 39.
buf 9-11.
char 9.
const 6.
cout 51.
cstdio 1.
d 10.
else 24.
end 45, 47-48, 50.
endl 51.
f 15, 17, 23.
for 50.
forward 15.
getline 38.
h 2.
i 50-51.
f
getline 38.
h 2.
i 50-51.
if 17, 20, 23, 25, 28, 31, 47.
in 34, 38.
include 0-2.
int 6, 15, 20, 45, 47, 50.
iostream 0.
is 23, 31.
istream 34.
key 15, 17, 23, 25, 31.
n 38.
namespace 4.
num 6, 10.
pager 2.
push 39.
read 34, 40.
return 12.
s 8, 11-12, 36, 38-39.
size 20-21, 47-48.
sprintf 10.
static 8.
std 4.
f
sprintf 10.
static 8.
std 4.
string 6, 8, 11, 36.
using 4.
void 15, 34, 43.
while 38.
f
b
iostream 0.
is 23, 31.
istream 34.
key 15, 17, 23, 25, 31.
n 38.
namespace 4.
num 6, 10.
pager 2.
push 39.
read 34, 40.
return 12.
s 8, 11-12, 36, 38-39.
size 20-21, 47-48.
sprintf 10.
static 8.
std 4.
string 6, 8, 11, 36.
using 4.
void 15, 34, 43.
while 38.
q
[ssdavis@lect1 p8]$
[ssdavis@lect1 p8]$ p8.out pager.cpp 1
#include <iostream>
```

```
#include <cstdio>
#include "pager.h"

using namespace std;

const string& Pager::IntToString(int num)
{
  static string s;
  char buf[100];
  sprintf(buf, "%d", num);
  s = string(buf);
  return s;
}  // IntToString()

void Pager::ProcessKey(int key)  // f = forward, b =
back
{
  if(key == 'f')
  {
    TopLine += 20;
    if(TopLine > (int) Text.size())
      TopLine = Text.size();
    WritePage();
f
    if(TopLine > (int) Text.size())
      TopLine = Text.size();
    WritePage();
  } // if key is 'f'
  else
    if(key == 'b')
    {
      TopLine -= 20;
      if(TopLine < 0)
        TopLine = 0;
      WritePage();
    } // if key is 'b'
}  // ProcessKey()

void Pager::read(istream &in)
{
  string s;

  while(getline(in, s, '\n'))
    Text.push_back(s);
}  // read()


f
}  // read()


void Pager::WritePage()
{
  int end = TopLine + 23;

  if(end > (int) Text.size())
    end = Text.size();

  for(int i = TopLine; i < end; i++)
    cout << Text[i] << endl;
} // WritePage()


f
b
{
  string s;

  while(getline(in, s, '\n'))
    Text.push_back(s);
}  // read()


void Pager::WritePage()
{
  int end = TopLine + 23;

  if(end > (int) Text.size())
    end = Text.size();

  for(int i = TopLine; i < end; i++)
    cout << Text[i] << endl;
} // WritePage()


b
void Pager::ProcessKey(int key)  // f = forward, b =
back
{
  if(key == 'f')
  {
    TopLine += 20;
    if(TopLine > (int) Text.size())
      TopLine = Text.size();
    WritePage();
  } // if key is 'f'
  else
    if(key == 'b')
    {
      TopLine -= 20;
      if(TopLine < 0)
        TopLine = 0;
      WritePage();
    } // if key is 'b'
}  // ProcessKey()

void Pager::read(istream &in)
{
  string s;

q
[ssdavis@lect1 p8]$
```