CSI 310 Data Structures          Spring 2004          Professor Chaiken
Project 2  Sequence Manager Using Linked Lists for Implementation  Due: March 3-5 (see project policy!)

Broken out instructions and some tips for Command 4 “newfirst”

Command> newfirst
Name: New York City, USA
Longitude (-180 to +180 degrees): -73.98
Latitude (-90 to +90 degrees): 40.77
(1) New York City, 73.98 degrees West, 40.77 degrees North.

Command> newfirst
Name: Cape Town, South Africa
Longitude (-180 to +180 degrees): 18.37
Latitude (-90 to +90 degrees): -35.92
(1) Cape Town, South Africa, 18.37 degrees East, 35.92 degrees South.

• The newfirst command makes the computer prompt for the input of the name and coordinates in the form exactly like the examples above.
  The command dispatcher would detect that the command is newfirst. It should then call some function(s) to input a new Landmark from the user and then insert this new Landmark into the list at the beginning.
  The prompts may be printed by a member or a constructor function of class Landmark.

• First the user must type the landmark name on one line of 80 characters or fewer, including the newline character.
  (This might be done by a member function of class Landmark)

• Next, the computer prompts for the longitude and latitude exactly as shown, and, for each, separately, the user must respond with a positive or negative number expressed in decimal, within the indicated ranges.
  (See below about checking the user input is within the indicated ranges. Tip: Omit range checking at first.)

• Negative signifies West longitude and South latitude.
  (This is important since the Landmark printing function prints “North”, “South”, “East” or “West” rather than signs of latitude and longitude measures.)

• Because of the modernistic style of simplicity, angle measurements will be expressed using decimal fractions of degrees, rather than the traditional degrees, minutes, and seconds.
  (Just read them with double x; cin >> x;)

• It would be OK for this project if the program refuses to go further until the input of a Landmark is complete. (So Landmark detail input can be done by the Landmark module. For simplicity, that module need not report input aborts or failures to its caller.)
(Again, implement the three prompts, for name, longitude, and latitude, and the reading of Landmark data, within member functions of class Landmark.)

- However, the following kind of error detection and recovery is required:

  Command> newfirst
  Name: Timbuktu, Mali
  Longitude(-180 to +180 degrees): -3.07
  Latitude(-90 to +90 degrees): 98.6
  Coordinate input out of range. Try again.
  Latitude(-90 to +90 degrees): 16.75
  (1) Timbuktu, Mali, 3.07 degrees West, 16.75 degrees North.

- After receiving the complete input of a Landmark, the computer responds by printing
  - first the position of the new Landmark in the sequence,
  - followed immediately by the name and coordinates.

  Note that the sign of each coordinate indicates whether the computer should print “East” or “West”, or “South” or “North”. By convention, 0 degrees is considered “East” or “North” according to which coordinate it represents.

  (You may have to write some “if” statements to implement this! There are other ways too.)

- The user, of course, must spell the command name newfirst exactly as shown, and your program must print the computer prompts (Name:, Longitude(-180 to +180 degrees):, etc. and the landmark report exactly as shown, except of course, for the data that the user had supplied.

- Input of the Landmark name must be read using the following code:

  (For simplicity, don’t bother removing any leading or trailing whitespace from it.)

```cpp
#include <iostream>
#include <cstdlib> // supplies exit()
using package std;
...
const int INBUFSIZE = 80;
char inbuf[INBUFSIZE];

if( !cin.getline(inbuf, INBUFSIZE) )
{
  cout << "ERR: Line input failed(too long?end of file?)..exiting." << endl;
  exit( 0 );
}
```

- The call to cin.getline() reads an input line up to INBUFSIZE characters long including the terminating newline character.
– The line includes any leading or trailing whitespace characters (spaces or tabs here).
– The cin.getline() call fails if the input is too long.
– This call stores the inputted characters into the given array except that it replaces the terminating newline with the NULL char value.
   That replacement turns the inputted characters into a C-string.
   Ask yourself and become sure of the answer: What is the maximum length of a line this software will handle?
– These rules apply even to the empty input line. Users will type that in by pressing the enter key only.
   It’s important to understand the empty C-string: It’s a NON-EMPTY array of the one character ’\0’.
– THE FOLLOWING IS A MISPRINT: If it is the empty line or a non-empty line not beginning with a colon, I’m sorry.. Just treat every input of the name the same.
– The program must determine its length and add 1 to the length in order to determine how long an array of char to dynamically allocate.
   (Why? What can happen if 1 isn’t added?)
– After dynamically allocating (using new[] of course..) the array, the line must be copied into the array.
   (Use strcpy() from the C-string library.. Look up in DSO.)

- The resulting new C-string will eventually become part of the data structure within instance of class Landmark.
  (So some code you write must copy the address returned by new char[length of name + 1] to the pointer-to-char data member in the new Landmark instance.)
- The input buffer inbuf above is NOT part of Landmark’s data structure!
  (If you do this, the names of all the Landmarks the program prints will be the same!)
- TIP: It would be really smart to design, implement and test a concrete class Landmark for most of the above functionality all by itself. See DSO pages 74-81 because a Landmark is essentially a fancy point with a name!