EECS 268: Spring 2009
Laboratory 3: Linked List ADT

Due: 11:59:59 pm the day before your lab meets the week of February 23

Lecture Topics: Linked Lists ADT and list operations
Lab Topics: Pointers, Dynamic Memory Management

1. Introduction

In this lab you will be implementing a simple linked list structure and defining methods to allow operations to be performed on this structure. CubeCompressor, a fledgling online movie rental company has asked you to implement the backend storage application for their web-based movie inventory system. Your system is to store a series of movies in a linked list structure. Your system must be able to insert, remove, swap, and print individual movies based on the position of the movie in the list. You will also need to be able to print the entire list in a formatted fashion. The overall purpose of this lab is to Understand and be able to implement the linked list abstract data type (ADT) in program developments.

2. Class Structures

You will implement two (2) classes for this project: MovieDatabase and Movie. They are described in detail below.

2.1. MovieDatabase

This class will define an ADT containing the pointers and structures necessary to hold a linked-list of Movies. This will include both a Node structure defining the linked-list node and a pointer to the head of the list. Pointers to movie instances will be stored in the node structure of the list.

Note: Rather than repeating Movie* throughout your ADT as the data type of your linked-list node item, you will need to create a new data type named NodeItemType using typedef. All references to the data type of your node item should use NodeItemType.

Here are some of the prototypes to the required methods for the MovieDatabase class:

```
      void insert(int index,NodeItemType newItem) throw
(MovieException);
      void remove(int index) throw (MovieException);
      void print(int index, std::ostream &out) throw (MovieException);
      void printAll(std::ostream &out);
      void swap(int pos1, int pos2) throw (MovieException);
```

insert – Given a position in the list and a pointer to a new instance of the Movie class, this method inserts the new movie into the list at position index.

> **Note**: The List index begins at position 1, so inserting into the list at position 1 would modify the very front of the list.

remove – Given a position in the list, this method attempts to remove the Movie at that position.

print – Given a position in the list, this method attempts to output the contents of the Movie instance at this position to the output stream out.

printAll – This method prints all Movies to the output stream.

swap – Given two positions, this method attempts to swamp the Movies stored in these nodes.

Given these required methods, you should investigate the use of a **find** method that returns a pointer to the ListNode at a particular index of the linked list.

2.2. Movie

Instances of this class will store information about a particular movie as well as methods to read in this information and print it out. Information you will need to store in the Movie class include:

- Name – the name of the movie
- Date – the year the movie was created
- Rating – R, PG, PG-13, etc.
- Description – one or two sentences describing the movie

For simplicity, you can store all these values as strings. These values should be read in from a file stream using the formatted input operator:

```
friend std::istream& operator>>(std::istream& in, Movie m);
```

And printed out using the << operator:

```
friend std::ostream& operator<<(std::ostream& out, Movie m);
```

3. Implementation Details

Exceptions are to be used for all error handling procedures. This will require a custom exception class to be created to be used when throwing and catching exceptions. The following situations are considered to be errors:

- Attempting to insert a movie into an invalid position.
- Attempting to delete a nonexistent movie.
- Attempting to print a nonexistent movie.
- Attempting to swap two movies where one or more of them is not present.

If an exception is thrown, an error message should be printed out indicating the cause of the error, and the program should continue processing input.

Dynamic memory allocation and deallocation should be used for each Movie created and each ListNode that the movie is stored in.

4. Input & Output

Your program will be expected to read input from the first file specified on the command line and to write output to the second file specified on the command line. Your program should display an error message and exit immediately if the expected command-line parameters are not provided.

Following is an example of how your program will be run from the command line:

```
$ ./lab3 input.txt output.txt
```

Input will be a series of commands, some of which are followed by data to be used with that command. Here are the commands your program must handle:

insert – This command will be followed by an integer representing the index on the list that the new movie will be inserted (remember, the list will use 1-based indexing). The following line will contain the information about the movie to be inserted. The values on this line will be delimited by the '|' character, and will be in the following order:

```
        <name>|<year>|<rating>|<description>|
```

Processing this command should involve creating a new Movie instance containing the appropriate information and attempting to add it to the linked list at the denoted position. Here is a full example of the insert command:

```
insert 1
Gamera|1965|G|A military plane crashes in the Arctic, awakening giant turtle-monster
Gamera.|
```

remove – This command will be followed by an integer representing the index of the movie to be removed. The processing of this command should involve attempting to remove the movie from the linked list at the denoted position. Example:

```
remove 2
```

print – This command will be followed by an integer representing the index of the movie to print. The processing of this command should involve attempting to output to file the information about the movie at the correct index in a formatted way. Example:

```
print 14
```

swap – This command will be followed by two integers representing the positions of the movies to be swapped. The processing of this command should involve utilizing the swap method to attempt to modify the positions in the list of the denoted movies. Example:

```
swap 1 3
```

print-all – This command will not be followed by any additional information. The processing of this command should involve outputting a detailed list of all the movies in the database and the current position of each movie. Example:

```
print-all
```

exit – This command serves as the terminating condition for your file reading. Processing this command should involve ending the application.

4.1. Sample Input File:

```
insert 1
Gamera|1965|G|A military plane crashes in the Arctic, awakening giant turtle-monster
Gamera.|
insert 2
Santa Claus Conquers the Martians|1964|G|In this children's movie, Martians abduct
Santa...|
insert 1
High School Big Shot|1953|PG|A high school dweeb plans a heist to win the favor of a
pretty classmate.|
print-all
insert 7
The Beast of Yucca Flats|1961|R|A defecting Russian scientist turns into a monster.|
print 2
remove 2
print-all
insert 3
Parts: The Clonus Horror|1979|PG-13|Clones are being bred to serve as a source of
replacement organs for the wealthy.|
swap 1 3
print-all
exit
```

4.2. Sample Output File:

```
Current Movies in the Database:
Position: 1
------------
High School Big Shot
       year: 1953
       rating: PG
       desc: A high school dweeb plans a heist to win the favor of a pretty classmate.

Position: 2
------------
Gamera
       year: 1965
       rating: G
       desc: A military plane crashes in the Arctic, awakening giant turtle-monster
Gamera.

Position: 3
------------
Santa Claus Conquers the Martians
       year: 1964
       rating: G
       desc: In this children's movie, Martians abduct Santa...

ERROR: Inserting at position out of range

Movie at position 2:
Gamera
       year: 1965
       rating: G
       desc: A military plane crashes in the Arctic, awakening giant turtle-monster
Gamera.

Current Movies in the Database:
Position: 1
------------
High School Big Shot
       year: 1953
```

```
        rating: PG
        desc: A high school dweeb plans a heist to win the favor of a pretty classmate.

Position: 2
------------
Santa Claus Conquers the Martians
        year: 1964
        rating: G
        desc: In this children's movie, Martians abduct Santa...

Current Movies in the Database:
Position: 1
------------
Parts: The Clonus Horror
        year: 1979
        rating: PG-13
        desc: Clones are being bred to serve as a source of replacement organs for the
wealthy.

Position: 2
------------
Santa Claus Conquers the Martians
        year: 1964
        rating: G
        desc: In this children's movie, Martians abduct Santa...

Position: 3
------------
High School Big Shot
        year: 1953
        rating: PG
        desc: A high school dweeb plans a heist to win the favor of a pretty classmate.
```

5. Grading Criteria

Grades will be assigned according to the following criteria:

30 **Implementation of Linked List**
- ⇒ Create necessary list structures
- ⇒ Implement required methods

20 **Implementation of Movie class**
- ⇒ Input and output methods
- ⇒ Data storage

15 **Exception Handling**
- ⇒ Correctly define new MovieException class
- ⇒ Properly throw exceptions
- ⇒ Properly catch exceptions

20 **Documentation**
- ⇒ Javadoc comments for all files, classes, and functions.
- ⇒ Internal documentation (especially recursive functions).

15 **Programming Style & Output Formatting**
- ⇒ Nice, intuitively formatted output and object oriented design practices.

100 **Total**