Dana Vrajitoru
C243 Data Structures

Homework 3

Due date: Tuesday, September 23, 2014. First phase: implement the queue class by Tuesday, September 16.

Ex. 1. a. Download the following files into a folder especially created for this assignment (don't mix them with the previous homework):
Makefile
general.cc
general.h
canfield.cc
canfield.h
Queue.h
Stack.cc
Stack.h
main.cc

Copy all the list related files from the previous homework, that is:
ListIterator.cc
ListIterator.h
ListNode.cc
ListNode.h
List.cc
List.h
These files should be the solution you have written for homework 2. If you need the solution for any of the functions used by the classes stack and queue, write me a message and I will provide them.

Add the following function to the List class:

```
// Very simple display of the List with no other messages.
void List::display()
{
    if (size)
    {
        ListIterator iter(head);
        while (iter)
        {
            cout << setw(2) << *iter << ' ';
            ++iter;
        }
    }
```

```
} // List::display()
```

You will also need to include the following header in the file List.cc:

```
#include <iomanip>
```

b. Create a file called queue.cc containing the implementation of the class methods for the class Queue based on the provided header file. Most of its functionality should be almost identical to the corresponding functions from the class Stack.

Compile the project with the command "make" and run the program with the command "canfield". At this point the program should compile but it won't run properly yet.

c. At this point the program should be functioning correctly. This program implements the game of Canfield. You can familiarize yourself with this game by running the program called AisleRiot that you can find in many Linux distributions. From the Game menu you can select New Game Of and then Canfield. The Help menu of this program will provide a set of rules for the game. A copy of them can be found here:

Canfield rules An online version (www.freeplaysolitaire.com/FreePlayCanfield.php)

Our program is a simplified version of the original Canfield in the sense that the cards are not marked with the suits, so the card movement rules are based only on numbers and not on colors. We also allow an unlimited amount of re-dealing.

Ex 2. Modify the Canfield game itself with the following improvements:

a. Implement another option in the first move screen where the user can choose to repeat the previous move. We can attach it to the character 'r'.

b. If a tableau is empty after a move and the reserve is not empty, a new card should automatically be moved to this tableau. You should do this both after removing a single card and a whole tableau.

c. After each movement, you should check if the player has won the game and display an appropriate message if this is the case. Add this feature as a function and find the most appropriate place to call it in the canfield.cc file.

d. Add a score keeping feature to the game that simply counts the number of cards moved to the foundations and display it below the cards.

e. Optional, for 3 extra credit points. Sometimes a whole tableau could be moved directly to a foundation. For this, if the user enters a 'T' to move a whole tableau and specifies a foundation as the destination, move the entire tableau to that foundation if possible. Note that the test for moving a whole tableau to a foundation is different from the

case where you move a tableau to another tableau and the card to be checked should be different in that case.

Notes.

- Upload to Oncourse all the files that you modify, meaning at least "queue.cc", "canfield.cc", and probably "canfield.h".
- If you need the solution for any of the functions required in the previous homework to do the new one, I can provide that by email (you need to send me a request first). However, you cannot take credit for those functions afterwards if you do so, meaning if you haven't turned in the previous homework yet.