CS 235 Midterm

Version 0.8

Instructors: K. Seamons and R. P. Burton

November 5 – 8, 2013 (Tuesday through Friday)

Due in the Lab on Friday no later than 6:00 p.m.

Penalty for submitting the midterm late:

30 points per day (including the weekend), advancing at 6:01 p.m. each day

Open Book (142 course text and your CS 235 course text only), Open Notes (including your own Lab solutions)

Open Secondary Storage Device: yours only

Open Laptop: if you wish

Open Course Website and www.cplusplus.com, but no other Internet resources (no Google)

Closed Neighbor (and everyone is thy neighbor)

**\*Instructions\***

(Please read carefully)

1. This midterm consists of a C++ programming problem with optional extra credit. Read and understand the statement of the problem completely before beginning to design, code, and test. As part of your design, consider the test cases that will establish the correctness of your solution. Test your solution thoroughly before submitting it.
2. Produce a solution, which consists of your C++ code, with a comment at the beginning of each file (both .h and.cpp) which includes your name, your CS 235 section, and “CS 235 Fall 2013 Midterm.” Upload your completed project by compressing the files and submitting through Learning Suite with TA assistance. If a packet is not collected by a TA upon submission, you will not be graded and will therefore receive no credit for the exam. Attribute any code taken from or based on other sources (excluding the course texts and the course websites). Attributed code copied from or based heavily on outside sources is worth half credit. Unattributed code copied from or based heavily on outside sources is worth no credit.
3. Understanding the problem correctly is part of the examination. If something seems unclear, ask a CS 235 TA for clarification. You may pose questions to the CS 235 TAs at any time. However, the TAs generally are not permitted to answer questions related to design, C++ implementation, debugging, or testing.
4. Prior to submitting your midterm, score it using the attached scoring sheet (this will help you maximize your points and will help us grade your exam accurately).
5. Your solution packet must all be stapled together before it will be accepted by a TA, even if this results in a late submission.
6. You will be given 5 points extra credit if your self-grading score is within 5 points of the TA Total score. You will be deducted 5 points if your self-graded score is more than 10 points different than the TA Total score.
7. When you are finished, submit your exam via Learning Suite with the assistance of a TA. An Agilix timestamp may be used when calculating the days late.
8. Sign the grading sheet to request that your midterm be graded and to certify that no unfair information related to the midterm has been received by you, either directly or indirectly, and that none will be conveyed by you. If we discover that you cheated or assisted someone in cheating, intentionally or unintentionally (including accidentally), your score for this exam may (and probably will) be rand() % 1.

We’re serious.

No Missing Links!

**Requirements**

* Extend the DoubleLinkedListInterface class to create a child class that is a double-linked list template class (see the hint below for the syntax to do this). The child class implements all of the method headers given for the DoubleLinkedListInterface abstract class found in DoubleLinkedListInterface.h.
* You must implement your own list. You may not use any predefined data structures from the STL (including Arrays).
* Submit an additional document with your solution stating the Big O of each member function. This must also include a brief explanation justifying the Big O stated for each member function.
  + Example:

for(int i = 0; i < n; i++)

{

for(int j = 0; j < n; j++)

{

//do something

}

}

The previous code is Big O(n2) because a double nested for loop will run the code in the outer loop n times, and the inner loop will run n times for each run of the outer loop. This is n\*n, or n2.

**Clarifications & Constraints**

* Your program needs to be “bullet-proofed”. You need to prepare for all possible input, even input which is not specifically mentioned in the specifications. This requires you as the programmer to anticipate what problems could arise from invalid input and to handle them appropriately.
* A test driver will be used to grade the exam; however it will not be available for student use. You should test your code thoroughly before submission.
* You are expected to create your own main function for testing purposes, but you will not be required to submit that file.
* Duplicate pieces of data should be allowed in your list.
* Your list must be a double-linked list. (A single-linked list will result in losing half of your total points; “ouch”).

**Hints & Tips**

* When inheriting from the DoubleLinkedListInterface class, refer to the following syntax. If your class were called DLL, you would write:

template <class T>

class DLL : public DoubleLinkedListInterface<T>{//class};

* You will not use a separate '.h' and '.cpp' file for your class that inherits from DoubleLinkedListInterface. Instead, you will need to implement, in the “.h” file of your class, all of the methods which inherit from DoubleLinkedListInterface. This is a special exception with template classes, which require the implementation to be in the '.h' file.

**Extra Credit** (precondition: submission of all required parts of the midterm. You may complete up to two of the following three extra credit options for extra points)

* (5 points) Operator Overloading. Overload the “==” (equals) operator for the class that inherits from DoubleLinkedListInterface.h.
* (5 points) Sorted List. Implement the sort function. You may not use any predefined C++ sort. Your list must be sorted in non-descending order.
* (5 points) Josephus Problem. You correctly solve the Josephus problem and return the correct safe index.

Josephus Problem Info:

The Josephus problem is named after the historian Flavius Josephus who lived between 37 and 100 BCE. Josephus was a reluctant leader of the Jewish revolt against the occupying Roman forces. When it appeared that Josephus and his band were going to be captured, they resolved to kill themselves. Josephus suggested, "Let us commit our mutual deaths to determination by lot. He to whom the first lot falls, let him be killed by him that hath the second lot, and thus fortune shall make its progress through us all; nor shall any of us perish by our own right hand, for it would be unfair if, when the rest are gone, somebody should repent and save himself." (Flavius Josephus, The Wars of the Jews, Book III, Chapter 8, Verse 7. Translated by William Whiston, 1737). As fate (or scheming) would have it Josephus was the last to be removed. He and the person he was to kill surrendered to the Romans. Josephus did not describe how the lots were assigned, but the following approach generally is believed to be the way it was done. Josephus and his band formed a circle. They counted around the circle to some predetermined number. When that number was reached, the lot "fell" on that individual and he left the circle. The count started over with the next person in the circle.

Your task for the Josephus Problem is to find the safe index and return that number. For arbitrary numbers m (m > 1) and n (10 <= n <= 50), compute where Josephus needed to stand to be the last one chosen; this location is referred to as the "safe index," the location that would be chosen last should the game be played with the specified m and n. You must use your linked list to calculate the solution to the problem. This means that the participants of the Josephus problem must be inserted into one of your lists, and an algorithm is performed using that list in order to calculate the safe index.

Josephus Examples:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** | **m** | **Players** | **Removal Order** | **Safe Index** |
| 20 | 1 | Josephus,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S | Josephus,A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S | 19 |
| 10 | 3 | Josephus,A,B,C,D,E,F,G,H,I | B,E,H,A,F,Josephus,G,D,I,C | 3 |
| 10 | 6 | Josephus,A,B,C,D,E,F,G,H,I | E,A,H,F,D,G,Josephus,I,C,B | 2 |

**Midterm Scoring Sheet** (A physical copy of this page must be turned in to the TAs)

Printed Name:**\_\_\_Eric Eyre\_\_\_\_\_\_** Professor Name:\_Seamons\_\_\_\_\_\_\_ Section #:\_1\_\_\_\_

Student Grading TA Grading

\_20\_\_/ 20 pts \_\_\_/ 20 pts – Templates

\_20/ 20 pts \_\_\_/ 20 pts – Creates a **template** class that inherits from the provided abstract class

\_10/ 10 pts \_\_\_/ 10 pts – Big O Analysis

10\_/ 10 pts \_\_\_/ 10 pts – A correct Big O analysis has been attached to the submitted documents

\_40/ 40 pts \_\_\_/ 40 pts – DLL Basic Functionality

15\_/ 15 pts \_\_\_/ 15 pts – Insertion

10\_/ 10 pts \_\_\_/ 10 pts – Removing

\_5\_/ 5 pts \_\_\_/ 5 pts – At

\_5/ 5 pts \_\_\_/ 5 pts – Contains

\_5\_/ 5 pts \_\_\_/ 5 pts – Size

\_40/ 40 pts \_\_\_/ 40 pts – DLL Intermediate Functionality

\_10/ 10 pts \_\_\_/ 10 pts – Swap

10\_/ 10 pts \_\_\_/ 10 pts – Shuffle

10\_/ 10 pts \_\_\_/ 10 pts – Count

10\_/ 10 pts \_\_\_/ 10 pts – Reverse

30\_/ 30 pts \_\_\_/ 30 pts – DLL Hard Functionality

10\_/ 10 pts \_\_\_/ 10 pts – Palindrome

10\_/ 10 pts \_\_\_/ 10 pts – Remove duplicates

10\_/ 10 pts \_\_\_/ 10 pts – kth from last

10\_/ 10 pts \_\_\_/ 10 pts – Coding Style

\_5\_/ 5 pts \_\_\_/ 5 pts – Neat Code: correct indentation, comments as needed, helpful variable names

\_5\_/ 5 pts \_\_\_/ 5 pts – No debugging cout statements or any print/pause statements in submitted files

**\_\_\_/ 10 pts \_\_\_/ 10 pts – Extra Credit**

\_\_\_/ 5 pts \_\_\_/ 5 pts – Operator Overloading

\_\_\_/ 5 pts \_\_\_/ 5 pts – Sorting

\_\_\_/ 5 pts \_\_\_/ 5 pts – Josephus Problem

\_\_\_/ 150 pts \_\_\_/ 150 pts – Total (before late penalties)

\_\_\_/ -30\*Days Late

\_\_\_/ 150 pts – Final Score \*(If your list is not double-linked, your Final Score will be halved)

Student to TA Comments:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Received By: TA Name) (Date Received) (Late) (Graded By: TA Name) (Date Graded)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, 2013

(Student Signature)\* (Date)

\*By signing above, you agree that you agree to all of the Instructions for this exam.