MIDTERM EXAM INSTRUCTIONS

1. Midterm Exam: **75 points w/ 5 E.C. points**
2. Due Date & Time: **11-12-2020 at 11:55 PM**

WHAT TO SUBMIT

1. Code
2. Assignment Report

HOW TO SUBMIT AND THE RULES TO FOLLOW

- Submit via iLearn, the Assignment Submission section
- Please refer to Assignment 01 for the Assignment Guidelines
- Please follow the Assignment Report Template
- Please follow the Course Policy on Student Conduct and Academic Honesty

PERFORMANCE TRACKER		
ASMT	**GRADE**	**YOUR GRADE**
ZOOM	05	
01	15	
02	100	
03	100	
MIDTERM 01	25	
04-PREPARATION	25	
04	75	
TOTAL	345	

A: 90-100% **B**: 80-89% **C**: 70-79% **D**: 60-69% **F**: 0-60%
The course grader provides feedback to your assignments on iLearn.

ABOUT

- Please download: http://csc340.ducta.net/Assignments/Assignment-04-Code.zip
- This assignment's three main topics are:
  - **Linked List** which was a topic of CSC 220 or of a previous course not at SFSU. We reviewed and learned Data Structures.
  - **Recursive Function** which was also a topic of CSC 220 or of a previous course not at SFSU.
  - **Smart Pointers** which we cover in detail in this course.
  - Assignment 04-Preparation helped us study the C++ versions of Linked List (the Linked Bag) and of Recursive Function. The focus was on C++ syntax and advanced implementations.
  - More help and sample code will be provided. Please start this assignment early.
- All parts of this assignment are to be done in C++.

PART A – Smart Pointers. **15 points**

- For each of the following statements, please:
  - Explain the statement in 5 or more sentences. Please think Interviews. And
  - Create a new code experiment to demonstrate our understanding.
  - *Please remember to submit our code and document our experiment in our assignment report.*

1. Deleting the same memory twice: This error can happen when two pointers address the same dynamically allocated object. If `delete` is applied to one of the pointers, then the object's memory is returned to the Free-store. If we subsequently delete the second pointer, then the Free-store may be corrupted.
2. Use smart pointers… Objects that must be allocated with `new`, but you like to have the same lifetime as other objects/variables on the Run-time stack. Objects assigned to smart pointers will be deleted when program exits that function or block.
3. Use smart pointers… Data members of classes, so when an object is deleted all the owned data is deleted as well (without any special code in the destructor).
4. Converting `unique_ptr` to `shared_ptr` is easy. Use `unique_ptr` first and covert `unique_ptr` to `shared_ptr` when needed.
5. Use `weak_ptr` for `shared_ptr` like pointers that can dangle.

PART B – Linked Bag. **40 points**

- Please change only files: `LinkedBag340.cpp` and `Include.h`, no other files.
- We are to implement 8 small additional functions and 2 helper functions to the Linked Bag.
- Our programs must produce **identical** output to the output in the 2 sample runs: `Asmt04_Run1.txt` and `Asmt04_Run2.txt`
  - Our Test 9's output must also be **identical** to the sample output excepts the random values.
  - Our Test 9's random values in our 2 sample runs' output must be **different**.

<u>Descriptions of the 8 functions:</u>
Please ask questions, if any, during the in-class discussions and demos for this assignment.

1. **`removeSecondNode340`** deletes the second node in the Linked Bag. **4 pts**
2. **`addEnd340`** inserts the new node at the end of the Linked Bag. **4 pts**
3. **`getCurrentSize340Iterative`** counts the number of nodes in the Linked Bag iteratively. **4 pts**
4. **`getCurrentSize340Recursive`** counts the number of nodes in the Linked Bag recursively. Use 1 helper function: **`getCurrentSize340RecursiveHelper`**. **4 pts**
5. **IMMEDIATE RECURSION: `getCurrentSize340RecursiveNoHelper`** counts the number of nodes in the Linked Bag recursively. This recursive function **does not** use any helper functions. **8 pts**
6. **`getFrequencyOf340Recursive`** recursively counts the number of times an entry appears in the Linked Bag. Use 1 helper function: **`getFrequencyOf340RecursiveHelper`**. **4 pts**
7. **IMMEDIATE RECURSION: `getFrequencyOf340RecursiveNoHelper`** recursively counts the number of times an entry appears in the Linked Bag. This recursive function **does not** use any helper functions. **8 pts**
8. **`removeRandom340`** removes a random entry from the Linked Bag. **4 pts**

PART C – Linked Bag, Smart Pointers Version. **20 points**

- Create a Smart Pointers version of our PART B's Linked Bag:
  1. Please create a copy of our entire PART B solution and name it: **`PartC_SmartPointers`**.
  2. Then go through all the files, not just **`LinkedBag340.cpp`**, and use smart pointers properly where it is possible.
  3. In our assignment report, **list the file names and the line numbers in which we use smart pointers. For each smart pointer, explain in 5 or more sentences why it is a proper use.**
  4. This Smart Pointers version must work properly and produce identical output like that of our PART B version.
  5. In addition, please **update** and **add destructor(s)** so that the program displays more information when object(s) get destroyed.
  6. Please remember to submit our code of this part. Save the code under a folder named "**`PartC_SmartPointers`**" and include this folder in the assignment submission ZIP. *Please remember to document this part in the assignment report.*

PART D – Linked Bag, Creativity. **5 Extra Credit points**

- Please create a copy of our entire PART C solution and name it: **`PartD_IamCreative`**
- This part is to show off our creative mind. Please implement a new function for Part C's LinkedBag. We need to add code to **`LinkedBag340.cpp`** and **`Include.h`** and write **`PartD.cpp`** to demonstrate how this new function works.
- **Requirements**, this function shall:
  1. Perform **one** meaningful task. Please use the first paragraph of at least 5 sentences in PART D to explain why it is a meaningful task.
  2. Modify the LinkedBag's content every time it runs.
  3. Use Smart Pointers in its parameter list, in its implementation, and as return value(s).
- Our graders expect higher quality in this part: creativity, a meaningful task, clean code, and clear documentation and report.

Happy coding and thank you!