NYU – Tandon School of Engineering

Brooklyn, NY 11201

CS-UY 2124 / OOP

Final Exam – 2 July 2020

Prof. Katz

Name: ____

NetID(first part of email):________________________________

- **YOU WILL TAKE THIS EXAM ON GRADESCOPE!!!**
- Please submit ONLY XXX files (one for each question)
- You can expect that the user inputs the appropriate values (int/float/etc where required).
- Comments are not required
- You may use your notes, your compiler and the Internet however you may not use any assistance from any other person (no Chegg resources, homework helper or another student)
- Anyone found cheating on this exam will receive a zero for the exam.
- If you have a question please email the professor (not the TAs)!

1. (20 pts) Write a recursive function which will receive a c-string (char*) and a character to search for. Your function cannot have any loops and should return the total number of occurrences of the character in the cstring. Your function signature must be as below:

   int findOccurrences(char* cstr, char toFind);

2. (45 pts) A simple hash table can be implemented, for integers, as an array (not vector) of linked lists. For a simple hash function, we can use the modulo to find the "hash" value of an integer. Specifically, if we have a 10 element array, and want to store the value 103, we would store it in position 3 (103%10=3). If we had a 15 element array, that same value would hash to 13 (103%15=13). Since multiple values can "hash" to the same location (collision), we use the linked list to resolve collisions (separate chain hashing).

   A good start is to have an array size of 5, but to double the size of the array (and rehash all of the values, of course) when the array gets more than 80% full (so the first resize will happen after 5 inserts).

   Please implement the following:

   - The IntHasher class which contains, at least, the following
     - Insert (add an int to the table)
     - Remove (take an int out of the table; do nothing if it doesn't exist)
     - Find (return true or false if it exists)
   - An output operator which prints ALL of the values in the hash table (order doesn't matter), one per line
   - An addition operator which allows for the combination of two intHasher objects.

3. (35 pts) We will be designing a management system for a national train system (think: Amtrak)! Our trains are made up of many cars, each of which could be a passenger car, or a cargo car. On any train, it is important that we know the weight of the train so as to be able to plan for fuel needed and stopping distances. Cargo cars have to account for the weight of all the cargo contained in them but passenger cars can make an estimate as to their weight based on the number of passengers multiplied by the weight of an “average” passenger (currently 220 pounds, including baggage). Please design the following classes and functions keeping in mind the restrictions above and below

   - Train. This class will store all of the cars in a vector and will have the following functions:
       i. an “addCar” function which adds one of the car types below
       ii. an “getTotalWeight” function which gets the total weight of all cars on the train
   - CargoCar. This class will store information about all of the cargo contained in the car. It will have the following functions:
       i. An “addPackage” function which will take a double, the weight of the cargo to be added to this car
       ii. A “getWeight” function which returns the total weight of the car (the sum of all packages)
   - PassengerCar. This class will store the number of passengers in the car. It will have the following functions:
       i. Overloaded ++ and - - operators (all of them) which will increase and decrease the number of passengers in the car
       ii. A “getWeight” function which returns the total weight of the car (the number of passengers multiplied by 220).
   - A main that creates a train with two passenger cars and a cargo car, adds a 1500 pound package to the CargoCar, adds a passenger to each of the PassengerCars and then prints the total weight of the train to the screen.

   Notes:
   - Use vectors to make your life easier!
   - You may inline functions as long as they do not exceed one line of code
   - Remember to use const where appropriate!