NYU, Tandon School of Engineering
Bridge to Computer Science Program — Winter 2022

Third Midterm Exam

*****PLEASE REMEMBER TO SUBMIT ON CLASSES, GRADESCOPE and EMAIL*******

- There are 100 points total.
- Note that there are longer programming problems at the end. Be sure to allow enough time for these.
- We supplied you with a file, named ‘solutions.txt’, where you should type all your answers in.
- For editing this file, you are allowed to use compilers such as Visual Studio, XCODE, CLion, textedit or notepad
- Calculators are not allowed.
- This is a closed-book exam. No additional resourced are allowed.
- Pay special attention to the style of your code. Indent your code correctly, choose meaningful names for your variables, define constants where needed, choose most suitable control statements, etc.
- In all questions you may assume that the users enter inputs as they are asked. For example, if the program expects a positive integer, you may assume that users will enter positive integers.
- No need to document your code in this exam, but you may add comments if you think they are needed for clarity.
- Read every question completely before answering it.

1. (3 pts) Which of the following sorting algorithms will have a consistent run time. In other words, which function's runtime can be directly calculated just by knowing the size of the vector and nothing more?

a) insertion sort
b) selection sort
c) quick sort
d) radix sort

2. (3 pts) Given "ptr" which is a pointer to an integer, which of the following is the correct way to output the address in memory where that pointer is stored

    a. cout<<ptr
    b. cout<<*ptr
    c. cout<<&ptr
    d. cout<<**ptr

3. (3 pts) We've designed a system in which there is a base class (call it Base) and a derived class which derive from Base (call it Derived). While inside the a member function of the derived class, what is the datatype of the "this" pointer?

    a. "this" has no datatype
    b. Base
    c. Derived
    d. The datatype cannot be determined

4. (3 pts) Sorting a linked list is difficult due to the lack of random access. One solution might be to remove the elements of the list, store them into a _____ temporarily and then reinsert them, one by one, back into the list.
    a. A Queue
    b. A Stack
    c. A Binary Search Tree

5. (3 pts) We would like to search a tree (note, it's not a BST) for a given value. I we believe that the element is, likely, closer to the root than the leaves, what would be a good traversal to use.

    a. Pre-order
    b. In-Order
    c. Post-order
    d. Level-Order

6. (15 pts) Given an, already, sorted vector of integers and an implementation of the Binary Search Tree class (BST<int>; note this is unbalanced), write a function which will insert the nodes from the vector into BST in a way that keeps the tree balanced.

7. (10 pts) A very large file (think many, many gigabytes) contains a large number of integers in sorted order, one per line. We would like to determine the median value of these integers. Explain, in English, how you would determine the median value (the median is the value that is in the middle of the file).

8. (15 pts) You are given a file, on the hard drive (file.txt; you are guaranteed that this file exists) which contains a set of random integers in unsorted order. Please design a function "findMedian" which will read in this file completely into memory (it fits, don't worry) and determine the median value. Your code will be evaluated for both efficiency and correctness.

9. (20 pts) A circularly linked list is one in which the last node's "next" pointer points back to the first node. In this type of list, none of the pointers, ever, point to null. You may assume that the LList class, the same one we created in class, has been slightly modified to create this type of list. Each node has only a single pointer, next, and no "tail" pointer exists in the class (only head). Please implement the push_back function for this class.

10. (25 pts) We are working for a shipping company and need a new system to track packages. Packages could be Envelopes or Boxes. Both have weight (a double; the weight in pounds) but Boxes, also, have cubicArea (a double) and so must be handled differently; we will keep track of Boxes and Envelopes separately.

    One of the most important aspects of our system is the ability to keep track of the "effectiveWeight" of the packages. Unfortunately, effectiveWeight and weight (on a measuring scale) are two different things. effectiveWeight of an Envelope is simply the weight, however effectiveWeight of a Box is the larger of its cubicArea and its weight.

    Design the three classes above using inheritance (every Box is a Package and every Envelope is a Package). You should provide constructors for all three classes which receive the relevant items (weight for an Envelope; weight and cubicArea for a Box). Provide a function in the Packages class which will return the effective weight.