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

Third Midterm Exam

- 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, VSCode, XCODE, CLion, textedit and notepad
- You may use 2 scratch papers.
- 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.

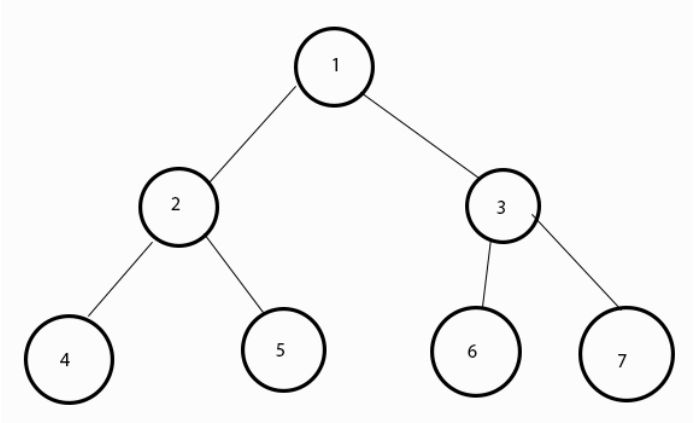
1. (3 pts) A queue is used to store items read in from a file. We will, after reading in all of the items, go through the queue and find the Nth element. How long will this whole operation take?
    a. Theta(1)
    b. Theta(N)
    c. Theta(N* log(N))
    d. Theta(N^2)
2. (4 pts) Given the code below, explain what is printed. If you believe there is an error, explain what is causing the error.

```
class Thing {
public:
        Thing(int newval=42) : {x = new int(newval); }
        int* x;
};

int main() {
        Thing one;
        cout << one->x << endl;
}
```

3. (8 pts) Given the following definition of a tree node:

```
class TreeNode {
public:
        int data;
        TreeNode* left;
        TreeNode* right;
};
```



Provide the code to print out all of the leaf nodes (only the leaves should be printed).

4. (5 pts) Given a linked list of integers, which sorting algorithm(s) below could we use to sort the entire structure (assuming you a limited in memory, you must do "in place" sorting)?
    a. InsertionSort
    b. MergeSort
    c. QuickSort
    d. Heapsort

5. (20 pts) A file on the hard drive contains numerous words separated by spaces. Please write a program to ask the user for the filename (do not continue until the user provides a name of a file that exists) and then read in all of the words in the file. Record the number of occurrences of each word and output the number of times each word appears. You do not need to sort the results, nor do you need to worry about punctuation or capital/lowercase. Only matching words should be counted. (Performance is not a consideration here)

    For example, if the file is: `the at word the word word at`

    The resulting output from your program should be:
    ```
    the 2
    at 2
    word 3
    ```

6. (15 pts) For this problem, you may assume we are using the stack class we explored in our webinar. Given a stack of integers, write a non-member function (you will only have access to the public functions) which will determine the number of times a given integer (passed as a parameter to your function) is found in the stack. You must do this in constant space, that is you may not duplicate the entire stack.

7. (10 pts) Describe, in English not code, how you would find the Nth element of an AVL tree. You may assume that the AVL tree node contains a height value. In your answer, provide the runtime of your algorithm.

8. (10 Points) Given an unbalanced binary search tree, provide the level order output of the tree if the insertions are (integers): 3,5,1,2,7,8

9. (25 pts) We've been asked to design a portion of a system to store information about pieces of technology equipment such as laptops, projectors and cell phones. For any of these devices we should store their serial number (a string provided on construction) but will also need to store some information specific to the device. For laptops, we will store their RAM quantity, for projectors we will store their bulb life and for cell phones we

will store their year of manufacturer. For all devices, we will need a function to get the serial number of the device.

Each different type has a different output when "printed" (we'll create a "print" function; no need to overload the output operator) but what it prints will be different for laptops, projectors and cell phones because each will print not only their serial number but also the items specific to their datatype. You should guarantee that only laptops, projectors and cell phones are printed, never any "generic" piece of equipment.

Part 1: Create classes for Laptops and Projectors (and any other classes necessary), you do not need to create a class for Cell Phones, someone else will do that. Make sure that the constructors take both the serial number AND the datatype specific material.

Part 2: Create a "Composite" class. Multiple devices can be connected together and we'd like to record that fact in the Composite class by retaining pointers to the items that are connected (please use a vector). Each will have it's own information, but the Composite class will have a function called "printItem(index)" which will cause the item at that index to print. A Composite item should overload the += operator to allow a new piece of tech equipment to be "added" to the vector.

Below, is a sample "main" function and the output from that function, to demonstrate how the classes are used.

```
int main() {
    Laptop dell("abc123", 8096);
    cout << "Dell: " << endl;
    dell.print();
    Projector epson("xyz34891", 10000);
    cout << "Projector:" << endl;
    epson.print();
    Composite together;
    together+= dell;
    together+=epson;
    cout << "**********************" << endl;
    cout << "Together: " << endl;
    together.printItem(0);

}
```

```
Dell:
Serial: abc123 ram: 8096
Projector:
Serial: xyz34891 bulb life: 10000
**********************
Together:
Serial: abc123 ram: 8096
```