NYU, Tandon School of Engineering
Bridge to Computer Science Program — Spring 2023

Third Midterm Exam

- You have two hours
- There are 100 points total.
- Note that there are longer 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.
- Write your name, netID and NYU ID at the head of the solutions file.
- For editing this file, you are allowed to use only plain text editors (Notepad for Windows users, or textEdit for Mac users).
- You are permitted to use Visual Studio (C++) or XCode as compilers. And Textedit/Notepad for text editing but should copy/paste your answers to the TXT file.
- 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.
- When done, please upload your answer file to Brightspace.nyu.edu, Gradescope and email to dkatz@nyu.edu

1. (5 pts) Given an object "o" of type Thing which contains a pointer to an int "ptr", which of the following would allow us to print the interger value to the screen?
   a. `cout<< o.ptr;`
   b. `cout<< o->ptr;`
   c. `cout<< *o.ptr;`
   d. `cout<< *ptr;`

2. (5 pts) Which data structure allows us to insert an arbitrary value in nearly Log N time?
   a. Array
   b. Linked List
   c. Stack
   d. Tree
3. (5 pts) Which of the following is an overloaded assignment operator for the Thing class?
   a. `Thing& operator=()`
   b. `Thing& operator(const Thing&)`
   c. `Thing& operator=(const Thing&) const`
   d. `Thing& operator=(const Thing&)`
4. (5pts) Given a pointer (ptr) to an object of type LList, a linked list with one head pointer (and no dummy nodes), provide a single line of code to print the first data value in the list. You can assume that each node contains a "data" variable and a "next" pointer.

5. (5 pts) In designing the Thing class, we make use of dynamic memory. To avoid problems such as garbage on the heap and double delete, which methods must be included in the Thing class?

6. (5 pts) Which method in the Stack class should be used for adding a new element onto the stack?

7. (10 pts) You are given a balanced binary search tree of N integers and would like to produce a vector of all values greater than a given value. Explain, in English, how you would accomplish this as efficiently as possible. (NB: you have access to the entire tree including all pointers)

8. (10 pts) Convert the following infix expressions to postfix form
    a. a*(b+c)-d
    b. a+b+c*d
    c. a-b*c*d

9. (20 pts) A file on the hard drive ("students.txt") contains student names, one per line. The names are not in alphabetical order, but we would like them to be. Please write a program to read in these names, sort them and store them back in the same file. For this problem, you may use any STL structure or function or you may simply assume that one of the sorting algorithms that we discussed are available (you do not need to write the sorting algorithm).

10. (30 pts) Our customer is a Coffee Shop which has asked us to design a new system for placing orders. We will be implementing the classes for this system. At the coffee shop orders can be placed for IcedCoffee or HotCoffee only (we're keeping it simple). All coffee items have a size (an integer number of ounces), cream (a boolean), and sugar (a Boolean). Iced coffees have an amount of ice (an integer, 5 as the default from 1-10) and Hot Coffees have a temperature (a double, 195.5 as the default). Each order can contain one of more coffees including both hot and iced possibly in the same order. In the future, we may add more types of Coffee to the order class (cappuccino anyone?) so your order class must accommodate all types of coffee, not just hot and iced.

    Ingredients are expensive, so you must provide a "cost" method in each class. The cost of an order is, of course, the sum of all of the costs of all of the coffees in that order. Use the following table to determine the price of the order:

Iced coffee base cost: $3.00 (additional costs below)

Size>32	$1.00
Add milk	$0.50
Add Sugar	$0.25
Ice >5	$0.50

Hot coffee base cost: $2.50 (additional costs below)

Size>16	$1.00
Add milk	$0.25
Add Sugar	$0.15
Temperature>210	$0.25

(continued on next page)

You should design the above classes including the HotCoffee, IcedCoffee and Order class as well as any other classes you might need. Below is a sample "main" to demonstrate how this works. You code MUST work with this below example.

```cpp
int main() {
    Order newOrder;
    HotCoffee h1(32, 190); //32 ounces, temperature 190
    h1.addSugar();
    HotCoffee h2(8, 215); //8 ounces, temperature 215
    h2.addCream();
    IcedCoffee i1(39, 7); //39 ounces, ice level 7
    i1.addCream();
    i1.addSugar();
    cout << "h1 cost: " << h1.cost() << endl; //Prints 3.65
    cout << "h2 cost: " << h2.cost() << endl; //Prints 3
    cout << "i1 cost: " << i1.cost() << endl; //Prints 5.25
    newOrder.addCoffee(&h1);
    newOrder.addCoffee(&h2);
    newOrder.addCoffee(&i1);
    cout << "Order total: " << newOrder.cost() << endl; //Prints 11.9
```