NYU, Tandon School of Engineering
Bridge to Computer Science Program — Winter 2024 Extended

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. (5 pts) When using an iterator on an STL list of Things (a class with only public data items), which of the below can be used to access the "name" variable in the Thing?
    a. itr.name
    b. itr->name
    c. itr->Thing->name
    d. *itr.name

2. (5 pts) Which data structure might be used to store items in a buffer where the first item received is the one which should be processed first?
    a. Array
    b. Linked List
    c. Stack
    d. Queue
3. (5 pts) A class that derives from another class is called a ?
    a. Base class
    b. Sub class
    c. Super class
    d. Abstract class

4. (5pts)A derived class wants to change the value in the integer "val." Provide the code which could be put in place of the comment which will change the value to 100.

```
class Base{
          int val;
    protected:
          void func(int x){ val = x; }
};

class Derived: public Base{
    public:
           void func(int x){/* Your code HERE!   */}
};
```

5. (10 pts) Given a pointer to an arbitrary node of a linked list, proved a function which will return which node number this node is. The "node number" is where this node would be found if it were in an array (i.e. the first node in the list is node number 0, the second is node number 1, etc).

6. (10 pts) Given an unsorted STL list of integers, provide an algorithm, in English, for determining if the STL list contains any duplicates. In your answer, please specify the runtime of your algorithm.

7. (10 pts) Given a pointer to the root of a BinaryTree we want to check the tree to make sure that all parent pointers are correctly set. Write a function to perform this check in an efficient manner. For reference, each node contains a "left", "right" and "parent" pointer.
8. (30 pts) A file on the hard drive ("input.txt" it exists, you do not need to test for it) contains two columns separated by a tab character. The first column is a student's grade (a float), the second is the student's NYU ID number (an integer). You are asked to create an output file (output.txt) with the ID numbers of all the students who are failing the class (i.e. those with a grade less than 65).
9. (25 pts) A PriorityQueue is a special type of Queue in which the "dequeue" function will always remove the lowest value from the Queue and not necessarily the oldest value from the queue. (Note: In most circumstances, we'd like the PriorityQueue to operate VERY efficiently, but that's not going to be the case for this question; efficiency doesn't matter). We have already defined the Queue class for you, it was given during lecture. You do not need to, and cannot, change anything in it! You may assume that whatever datatype being stored has overloaded the function "operator<" which is adequate for comparisons.

   For this question, please derive a new class from Queue called PriorityQueue. Override any necessary functions to meet the above requirements.

```
int main() {
    PriorityQueue<int> pq;
    pq.enqueue(100);
    pq.enqueue(20);
    pq.enqueue(50);
    pq.enqueue(35);
    cout << pq.top(); //prints 20
    cout << pq.dequeue(); // prints 20
    cout << pq.dequeue(); //prints 35
    //The PriorityQueue would still contain 50 and 100.

}
```