CSc 127B Exam 2 Practice Test

The best way to use this exam to study is to try and take the test once without any notes. Then, use the questions you struggled with or got wrong to decide what to study. However, please note that, while the SLs do their best to write questions that will reflect the test, there may be some things that are not on this practice test that appear on the real test. So, make sure to study your notes and the lectures in addition to taking this!

1. Fill in the blanks for the following
   a) In a stack, the first thing put on the stack is the ___________ thing taken out.
   b) The three basic functions of a stack are: ____________, ____________ and ____________.
   c) In a queue, the first thing put in the queue is the ___________ thing taken out.
   d) The two basic functions of a queue are: ____________ and ____________
   e) A queue which partially sorts the data enqueued based on its importance is called a ________________.

2. We learned a bit about stacks and queues in class. Below is the output of a mystery class after various functions have finished executing. PutOn will put an int onto the stack or queue and TakeOff will take the int off the stack or queue.

   a) Based on the toString is the mystery class a stack or a queue?

      Input:
      PutOn(7); putOn(10); putOn(42); takeOff(); takeOff(); putOn(7);
      putOn(7); System.out.println(toString());
      Output: 7 7 7

   b) What is a potential error you would want to check for when using this class?
3) Using a ArrayList, finish this unknown class.

```java
public class Unknown {
    public ArrayList<Integer> arrayList = new ArrayList<>();
    public Unknown() {
    }

    // Write putOn

    // Write takeoff

    public String toString() {
        StringBuilder str = new StringBuilder();
        for(int i = 0; i < arrayList.size(); i++) {
            str.append(arrayList.get(i) + " ");
        }
        return str.toString();
    }
}
```
4. Given a base (denoted by n) and a coefficient (denoted by k), where n,k are integers, write a recursive method called exponent(int, int) that returns $n^k$. You can assume that $n > 1$.

5. What is the default size if you do not specify a size when you declare an ArrayList?
6. Given an arbitrary integer value, print recursively all odd numbers down till 1.
Ex. Given 5, your program should print:
5
3
1
You can assume k > 0

7. Given a string, recursively compute a new string where all the lowercase 'x' chars have been moved to the end of the string.
endX("xxre") → "rexx"
endX("xxhixx") → "hixxxx"
endX("xhixhix") → "hihixxx"
8. Given the following block of code, What does the following code print?

```java
Stack stack = new Stack();
Queue queue = new Queue();

queue.enqueue(a);
queue.enqueue(b);
queue.enqueue(c);
queue.enqueue(d);

stack.push(queue.dequeue());
stack.push(queue.dequeue());
stack.push(queue.dequeue());
stack.push(queue.dequeue());

while(!stack.isEmpty()){
    System.out.println(stack.pop());
}
```

9. Short Answer: Which would you use to represent a stack/queue, an array or a linked list? For whichever answer you suggest, give some reasoning as to why you chose an array over a linked list or vice versa.
10. Given a nonempty linked list, remove the head of the linked list and return the value.

method header is:
public T removeHead(Node head)

Assume we are giving you a getNext method, getData method, setNext method, and setData method.

11. Suppose I have the following code for my Node class.

public class Node<E> {
    private E name;
    private Node<E> next;

    public Node (E element) {
        name = element;
        next = null;
    }
}

Suppose my linkedList class needs to sort the nodes in alphabetical order. Create a method call insert that will place a new node into the correct order. Make sure that you cover all edge cases and use your generics appropriately. You can assume this is a linked list is not circular and is a singly linked list. You can also assume in linkedList there is a variable that keeps track of the amount of items stored in the list called size. Also assume the compareTo method works correctly as well. Assume there is a head variable. It may or may not be set. Make this method return the index where the node was inserted at. There are to be no duplicates. If the name already exists in the list return -1.
12. Now write the compareTo method for the insert problem above. Return 1, if greater than, -1 if less than, or 0 if equal.
a) What is the difference between a doubly linked list and a singly linked list?

b) Why might you want to use a doubly linked list instead of a singly linked list?

c) Write a method that prepends to the front of a doubly linked list. Don’t forget to take care of any error conditions you can think of! Note: assume the DLLNode class contains the following: getNext(), setNext(), getPrev(), setPrev(), DLLNode(String data).

```java
public class DLList{
    DLLNode head = new DLLNode(null);

    public void prepend(String data){
        DLLNode addMe = new DLLNode(data);
    }
}
```