CSCI 151  
Exam 1  
November 5, 2021

The exam has 6 numbered questions. Questions 1 and 4 are worth 20 points each and the other four questions are worth 15 points each, for a total of 100 points.

This is a closed-book, closed-notes, closed-Internet exam.

After you have finished the exam please indicate whether you followed the Honor Code on the exam.

I □ did □ did not

adhere to the Honor Code while taking this exam.

__________________________________  Signature
1.

A. Suppose S is a Stack<Integer> that starts off empty and we do the following sequence of operations:

```java
S.push(1);
S.push(2);
S.pop();
S.push(3);
S.pop();
S.push(4);
S.push(5);
```

With the same stack we do the following loop:
```java
while (!S.isEmpty()) {
    System.out.print(S.pop());
}
```

What will this loop print?

B. Next, we do this with a queue. Suppose Q is an empty queue and we do
```java
Q.enqueue(1);
Q.enqueue(2);
Q.dequeue();
Q.enqueue(3);
Q.enqueue(4);
Q.dequeue();
Q.dequeue();
Q.enqueue(5);
```

What will the following loop print?
```java
while (!Q.isEmpty()) {
    System.out.print( Q.dequeue() );
}
```
2. Here is a loopy chunk of code:

```java
int total = 0;
for (int i = 0; i < N; i++) {
    total = total + i*i;
    for (int j = N-1; j >= 0; j--) {
        total = total - j;
        for (int k = 1; k < 10; k++) {
            total = total + k;
        }
    }
}
for (int m = 0; m < N; m++) {
    total = total + 1;
}
System.out.println(total);
```

Give a Big Oh upper bound for the runtime of this in terms of N.
3. What good are interfaces? **Describe** (in 3 sentences or less) **any situation where it is useful to have an interface.**
4. In Lab 3 you implemented Queues with a linked structure. Suppose we use an ArrayList instead, as we did with Stacks. Here is a start:

```java
public class MyQueue<T> implements QueueADT<T> {
    ArrayList<T> data;
    int size;
    public MyQueue() {
        data = new ArrayList<T>();
        size = 0;
    }
    ...
}
```

A. Write `public void enqueue(T item)` for this implementation.

B. Write `public T dequeue()` throws `NoSuchElementException` for this implementation.
5. For this question I will make a new data structure that I call a BobList<T>.
   A BobList is just like an ArrayList, with the addition of a method removeRandom()
   which removes and returns a random element of the list:

   ```java
   public class BobList<T> extends ArrayList<T> {
       T removeRandom() {
           Random rand = new Random();
           int i = random.nextInt(size());
           return remove(i);
       }
   }
   ```

   Now think about the Maze lab. I want to make a new MazeSolverBob class that uses a
   BobList<Square> as its worklist, with the BobList add() and removeRandom() for its
   add() and next() methods. In other words, we’ll store the worklist as a list, and each
time we take an element from it we will get a random element of the worklist rather
than the first or last element.

   Here is the question: **will this still solve the maze?** Why or why not?
6. **Write method** `sums(L)` **whose signature is**

   ```java
   ArrayList<Integer> sums(ArrayList<Integer> L)
   ```

   This method takes a list `L` as an argument and it returns a list that I will call `M`. The first entry of `M` is the first entry of `L`. The second entry of `M` is the sum of the first two entries of `L`. The third entry of `M` is the sum of the first three entries of `L`, and so forth. For example, if `L` has entries 3 5 7 then the list returned by `sums(L)` will have entries 3 8 15.