

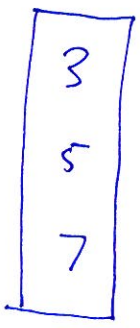
Question 1. [8 points] What output is printed by the following code?

```
Stack<Integer> s = new Stack<Integer>();  
s.push(3);  
s.push(5);  
s.push(7);  
System.out.println(s.pop());  
s.push(9);  
System.out.println(s.pop());  
System.out.println(s.pop());
```



Question 2. [8 points] What output is printed by the following code?

```
Queue<Integer> s = new LinkedList<Integer>();  
s.add(3);  
s.add(5);  
s.add(7);  
System.out.println(s.remove());  
s.add(9);  
System.out.println(s.remove());  
System.out.println(s.remove());
```



Question 3. [8 points] Briefly explain (a) the bug in the following code, and (b) how to fix the bug.

```
public static int sumIntCollection(Collection<Integer> c) {  
    int sum = 0;  
    for (int i = 0; i < c.size(); i++) {  
        sum += c.get(i);  
    }  
    return sum;  
}
```

not a list!

Collection doesn't include the get(int) method: need to use an iterator or foreach loop, e.g.

```
for (Integer val : c) {  
    sum += val;  
}
```

Question 4. [8 points] Complete the following method. It should return a list containing the same elements as the list passed as the parameter, but in reversed order. **Requirement:** Use a stack to reverse the elements of the original list.

Hints:

- Think about how a stack will help you reverse the elements of the original list
- Don't modify the original list: create a new empty list (for example, and `ArrayList`), add the reversed elements to it, and return it as the result of the method
- The method is generic, and the list elements have type `E`

```
public static<E> List<E> reverseList(List<E> orig) {  
    // Use this stack to help reverse the list  
    Stack<E> s = new Stack<E>();  
    for (E val : orig) {  
        s.push(val);  
    }  
    List<E> result = new ArrayList<E>();  
    while (!s.isEmpty()) {  
        result.add(s.pop());  
    }  
    return result;  
}
```

Question 5. [8 points] What output is printed by the following code? Explain your answer briefly.

```
Set<String> set = new HashSet<String>();  
set.add("A");  
set.add("B");  
set.add("C");  
set.add("D");  
set.add("A");  
set.remove("B");  
System.out.println(set.size());
```

at this point, the set contains A, B, C, D, size is 4
B is removed
prints 3 because remaining elements are A, C, D

Question 6. [8 points] Complete the following method. It takes two parameters: a map called `prices`, and a list called `order`. The `prices` map associates names of items with prices (in cents). The `order` list contains a list of items. The method should return the total price of the order, using the map to look up the price of each item in the list.

Here is example code which shows the behavior of the method:

```
Map<String, Integer> examplePrices = new HashMap<String, Integer>();
examplePrices.put("apple", 80);
examplePrices.put("orange", 120);
examplePrices.put("lemon", 75);
examplePrices.put("persimmon", 200);
List<String> exampleOrder = Arrays.asList("apple", "apple", "orange", "lemon");
int exampleTotal = orderTotal(examplePrices, exampleOrder);
System.out.println(exampleTotal); // prints 355 (which is 80+80+120+75)
```

Hints:

- Use a loop to iterate over the elements of the `order` list
- You can assume that the `prices` map is guaranteed to have an entry for each element in the `order`

```
public static int orderTotal(Map<String, Integer> prices, List<String> order) {
```

```
    int total = 0;
    for (String item : order) {
        total += prices.get(item);
    }
```

```
    return total;
```

```
}
```

Question 7. [8 points] Complete the following method so that it returns the sum of the digits of the integer n given as its parameter. You may assume that n is non-negative. For example:

- `sumDigits(2)` should evaluate to 2
- `sumDigits(12)` should evaluate to 3
- `sumDigits(90125)` should evaluate to 17

Important: Your implementation must be recursive. Do not use a loop.

Hint: Think of an appropriate base case.

Hint: For any non-negative integer n , $n \% 10$ is the value of the rightmost digit, and $n / 10$ is an integer containing all of the digits of n except for the rightmost digit.

```
public static int sumDigits(int n) {  
    if (n < 10) {  
        // Base case, 1-digit number  
        return n;  
    } else {  
        return sumDigits(n / 10) + (n % 10);  
    }  
}
```