1. Assume that Stack and Queue are implementations of the corresponding ADTs using Java generics. Consider the following method

```java
public static void problem1(Stack<E> s){
    Queue<E> q = new Queue<E>();
    while (! s.empty())
        q.enqueue(s.pop());
    while (! q.empty())
        s.push(q.dequeue());
}
```

(a) Let stack be a stack of Integers containing the data

```
7
10
5
-2
```

How is stack changed (if at all) when problem1(stack) is called? Trace through the method carefully.
Solution: For the first loop, we have the following:

```
7 10 5 -2
---
s q: (empty)
10 5 -2
---
s q: 7
5 -2
---
s q: 7 10
-2
---
s q: 7 10 5
---
s q: 7 10 5 -2
```
Then the second loop does the following:

```
  7
  ___
  s   q: 10 5 -2

  10
  7
  ___
  s   q: 5 -2

  5
  10
  7
  ___
  s   q: -2

  -2
  5
  10
  7
  ___
  s   q: (empty)
```

(b) If `st` is any stack, what is the effect of calling `problem1(st)`? Justify your answer.

**Solution:** `problem1(st)` will reverse the order of the elements of the stack. (Note since `st` is an `Object`, and not a primitive data type, it is passed by reference into the method, and thus the method can change its contents.)

2. Trace through the state of the stack `s` in the following code fragment.

```java
Stack<String> s = new Stack<String>();
s.push("happy");
s.push("sad");
String st = s.peek();
s.push("numb");
s.push(st+"dle");
```
s.pop();
st = s.pop();
s.push(st);

Solution:

3. Trace though the state of the queue q in the following code fragment.
(Assume Queue is an implementation of the standard queue interface using java generics.)

Queue<Integer> q = new Queue<Integer>();
q.enqueue(5);
q.enqueue(7);
q.enqueue(13);
q.dequeue();
Integer t = q.peek();
q.enqueue(12+t);
q.dequeue();
q.enqueue(q.dequeue());

Solution:

q: (empty)
q: 5
q: 5 7
q: 5 7 13
q: 7 13

t: 7
q: 7 13 19
q: 13 19
q: 19 13
4. Evaluate the following postfix expression

\[
3 \ 5 \ 7 \ 2 \ 8 \ + \ - \ + \ 4 \ -
\]

Show your work.

**Solution:** Work from the leftmost operator

\[
\begin{align*}
3 & \ 5 \ 7 \ 2 \ 8 \ + \ - \ + \ 4 \ - \\
3 & \ 5 \ 7 \ 10 \ - \ * \ + \ 4 \ - \\
3 & \ 5 \ (-3) \ * \ + \ 4 \ - \\
3 & \ (-15) \ + \ 4 \ - \\
& \ (-12) \ 4 \ - \\
& \ (-16)
\end{align*}
\]

So the answer is \(-16\).

5. Fully parenthesize the following Java expression (using the standard Java rules of precedence of operations and left-right associativity).

\[
x + 3 \ / \ ( \ y \ * \ 2 \ - \ 4) \ * \ w \ - \ 1
\]

**Solution:**

\[
( ( x + ( ( 3 \ / \ ( ( y \ * \ 2 ) - 4) ) * w ) ) - 1 )
\]

6. What is the output to the screen?

```java
public class Problem6{
    public static void main (String[] args){
        try{
            for (int i=1; i<=10; i++)
                System.out.println(g(i));
        }
        catch (Exception e){
            System.out.println("Exception caught.");
        }
        int[] ar = new int[10];
        for (int i=0; i<10; i++)
```
ar[i] = i*i;
ar[10] = 500;
System.out.println("End of main.");
}
public static int g(int k){
    return 120/(7-k);
}
}

Solution:

20
24
30
40
60
120
Exception caught.
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 10
    at Problem6.main(Problem6.java:13)

7. Consider the language whose sentences are given by $\langle W \rangle$

$\langle W \rangle = \langle W \rangle \ t \mid t \mid \langle W \rangle \ \langle S \rangle$
$\langle S \rangle = a \mid b \mid c$

Write a Java method

public static boolean inW (String s)

which determines whether the string $s$ is in the language of $W$. (Recall
that the method call $s$.substring($i$, $j$) returns the substring of $s$
going from index $i$ to index $j-1$ inclusive.)

Solution:

public static boolean inW (String s){
    int len = s.length();
    if (len==0)
return false;
else if (len==1)
    return s.charAt(0)=='t';
else{ // len>1: recursive case
    char last = s.charAt(len-1);
    if (last=='t' || last=='a' || last=='b' || last=='c')
        return inW (s.substring(0,len-1));
    else
        return false;
}

8. Consider an implementation StackReferenceBased which implements the StackInterface, uses Java generics, and throws a StackException. The implementation uses a linked list structure with top as the head of the list.

Write the code for the method

    public E pop() throws StackException

The class StackException is given by

    public class StackException extends RuntimeException{
        public StackException (String s){
            super(s);
        }
    }

Solution:

    public E pop() throws StackException{
        if (top==null)
            throw new StackException("attempt to pop off an empty stack");
        else{
            E ret = top.getItem();
            top = top.getNext();
            return ret;
        }
    }