Prob. 1 / 1

a - b * c / d * e ^ f + g

1	2	3	4	5	6	7	8	9	10	11	12	13
A	-	B	*	C	/	D	*	E	^	F	+	g

#	Symbol	Stack	post
1	A		A
2	-	-	A
3	B	-	A B
4	*	- *	A B
5	C	- *	A B C
6	/	- /	A B C *
7	D	- /	A B C * D
8	*	- *	A B C * D /
9	E	- *	A B C * D / E
10	^	- * ^	A B C * D / E
11	F	- * ^	A B C * D / E F
12	+	+	A B C * D / E F ^ * -
13	G	+	A B C * D / E F ^ * - G
14			A B C * D / E F ^ * - G +

Prob. 1 / 2

6 5 2 ^ 2 3 + 8 * - 3 - *

1	2	3	4	5	6	7	8	9	10	11	12	13
6	5	2	^	2	3	+	8	*	-	3	-	*

#	Task	Stack	
1	6	6	
2	5	6,5	
3	2	6,5,2	
4	^	6,25	2 ^ 5 = 25
5	2	6,25,2	
6	3	6,25,2,3	
7	+	6,25,5	2 + 3 = 5
8	8	6,25,5,8	
9	*	6,25,40	5 * 8 = 40
10	-	6,-15	25 – 40 = -15
11	3	6,-15,3	
12	-	6,-18	-15 - 3
13	*	-108	6 * -18

Prob. 1 / 3

6 5 2 ^ 2 3 + 8 * - 3 - *

1	2	3	4	5	6	7	8	9	10	11	12	13
6	5	2	^	2	3	+	8	*	-	3	-	*

#	Symbol	Stack (infix)
1	6	6
2	5	6,5
3	2	6,5,2
4	^	6,(5 ^ 2)
5	2	6,(5 ^ 2),2
6	3	6,(5 ^ 2),2,3
7	+	6,(5 ^ 2),(2+3)
8	8	6,(5 ^ 2),(2+3),8
9	*	6,(5 ^ 2),((2+3) * 8)
10	-	6,((5 ^ 2) - ((2+3) * 8))
11	3	6,((5 ^ 2) - ((2+3) * 8)),3
12	-	6,(((5 ^ 2) - ((2+3) * 8)) - 3)
13	*	6 * (((5 ^ 2) - ((2+3) * 8)) - 3)

<u>Prob. 1 / 4</u>

5 + 6 ^ 2 / 2 / 3 - 2 * 4 * 7

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
5	+	6	^	2	/	2	/	3	-	2	*	4	*	7	

#	Symbol	Stack (operand)	Stack (Operation)	
1	5	5		
2	+	5	+	
3	6	5,6	+	
4	^	5,6	+ ^	
5	2	5,6,2	+ ^	
6	/	5,36	+ /	6 ^ 2
7	2	5,36,2	+ /	
8	/	5,18	+ /	36/2
9	3	5,18,3	+ /	
10	-	11	-	18/3 = 6, 5+6 = 11
11	2	11,2	-	
12	*	11,2	- *	
13	4	11,2,4	- *	
14	*	11,8	- *	2*4
15	7	11,8,7		8 * 7 = 56, 11-56 = 45

Prob. 2 / 1

```
public static<T> void removeLast(LinkStack<T> st)
{
     LinkStack<T> temp = new LinkStack<T>();

     while(! st.empty())
          temp.push(st.pop());

     if(! temp.empty())
          temp.pop();

     while(! temp.empty())
          st.push(temp.pop());
}
```

Prob. 2 / 2

```
public static <T> boolean topEqualsBottom(LinkStack<T> st)
{
     if (st.empty())
          return false;

     LinkStack<T> temp = new LinkStack<T>();
     T last = null,first;

     first = st.pop();
     temp.push(first);

     while(! st.empty())
          temp.push(st.pop());

     if(! temp.empty())
     {
          last = temp.pop();
          st.push(last);
     }

     while(! temp.empty())
          st.push(temp.pop());

     return last == first;
}
```

Prob. 3 / 1

```
public static boolean containsMult3(int a[], int n)
{
     if (n < 0)
          return false;
     else if (a[n] % 3 == 0)
          return true;
     else
          return containsMult3(a, n-1);
}
```

Prob. 3 / 2

```
public static boolean sameSign(int a[], int n)
{
     if (n >= 0)
     {
          if (a[n] > 0 && a[a.length-1] > 0 ||
             a[n] < 0 && a[a.length-1] < 0)
               return sameSign(a, n-1);
          else if (a[n] >= 0 && a[a.length-1] <= 0 ||
             a[n] <= 0 && a[a.length-1] >= 0)
               return false;
     }

     return true;
}
```

Prob. 4 / 1

```
   public boolean search(T k)
    {
     return recSearch(head, k);
    }

    private boolean recSearch(Node<T> p, T k)
    {
     if(p == null)
          return false;

     if(p.data.equals(k))
          return true;

     return recSearch(p.next, k);
    }
```

Prob. 4 / 2

```
    public void reverse()
    {
     T x = null;

     if (! empty())
     {
          x = pop();

          reverse();
          top++;
          nodes[maxsize - top - 1] = x;
     }
    }
```