Question1:

Consider the following sequence of stack operations:

push(d), push(h), pop(), push(f), push(s), pop(), pop(), push(m).

(a) Assume the stack is initially empty, what is the sequence of popped values, and what is the final state of the stack? (Identify which end is the top of the stack.)

(b) Suppose you were to replace the push and pop operations with enqueue and dequeue respectively. What would be the sequence of dequeued values, and what would be the final state of the queue? (Identify which end is the front of the queue.)

Solution 1:

(a) Sequence of popped values: h,s,f. State of stack (from top to bottom): m, d

(b) Sequence of dequeued values: d,h,f. State of queue (from front to back): s,m.

Question 2:

Suppose you have a stack in which the values 1 through 5 must be pushed on the stack in that order, but that an item on the stack can be popped at any time. Give a sequence of push and pop operations such that the values are popped in the following order:

(a) 2, 4, 5, 3, 1

(b) 1, 5, 4, 2, 3

(c) 1, 3, 5, 4, 2

It might not be possible in each case.

Solution 2:

24531	15423	13542
push 1	push 1	push 1
push 2	pop	pop
pop	push 2	push 2
push 3	push 3	push 3
push 4	push 4	pop
pop	push 5	push 4
push 5	pop	push 5
pop	pop	pop
pop	x	pop
pop	(not possible)	pop

Question 3:

Use a stack to test for balanced parentheses, when scanning the following expressions. Your solution should show the state of the stack each time it is modified. The "state of the stack" must indicate which is the top element.

Only consider the parentheses [,], (,), {,}. Ignore the variables and operators.

(a) **[a+{b/(cd)+e/ (f+g)}-h]**

(b) **[a{b+[c(d+e)-f]+g}**

Solution 3:

```
(a) -        means empty stack
    [
    [{
    [{(      TOP OF STACK IS ON THE RIGHT
    [{
    [{(
    [{
    [
    -        empty stack, so brackets match
(b) -
    [
    [{
    [{[         TOP OF STACK IS ON THE RIGHT
    [{[(
    [{[
    [{
    [           stack not empty, so brackets don't match
```

Question 4:

(a) Suppose you have three stacks s1, s2, s2 with starting configuration shown on the left, and finishing condition shown on the right. Give a sequence of push and pop operations that take you from start to finish. For example, to pop the top element of s1 and push it onto s3, you would write s3.push(s1.pop()).

(b) Same question, but now suppose the finish configuration on s3 is BDAC (with B on top)?

Solution 4:

(a)
```
s2.push( s1.pop() )
s2.push( s1.pop() )
s3.push( s1.pop() )
s3.push( s1.pop() )
s3.push( s2.pop() )
s3.push( s2.pop() )
```

(b)
```
s2.push( s1.pop() )
s2.push( s1.pop() )
s3.push( s1.pop() )
s1.push( s2.pop() )
s3.push( s2.pop() )
s2.push( s1.pop() )
s3.push( s1.pop() )
s3.push( s2.pop() )
```

Question 5:

Consider the following sequence of stack commands:

push(a), push(b), push(c), pop(), push(d), push(e), pop(), pop(), pop(), pop().

(a) What is the order in which the elements are popped? (Give a list and indicate which was popped first.)

(b) Change the position of the pop() commands in the above sequence so that the items are popped in the following order: b,d,c,a,e.

You are not allowed to change the ordering of the push commands.

Solution 5:

(a) c (popped first), e, d, b, a

(b) push(a), push(b), pop(), push(c), push(d), pop(), pop(), pop() push(e), pop()