Stack - a Last In First Out container object

All insertions and deletions from the top

examples

1) RAM addresses when recursion rewinds

2) Quickly reverse

3) Directory management

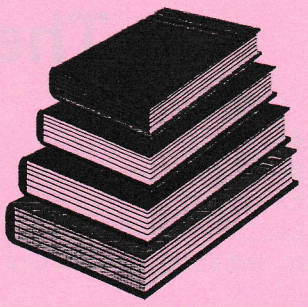
4) Reverse Polish Notation

When writing Stack code, it matters whether:

1) At client file, in which case, Stack object push, pop, top is empty?

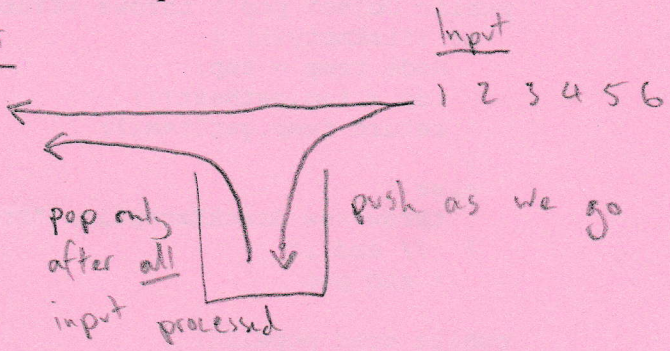
2) At implementation level, in which case, must work with Stack linked list with one pointer to the top to a non-circular linked list

Stack Exercises

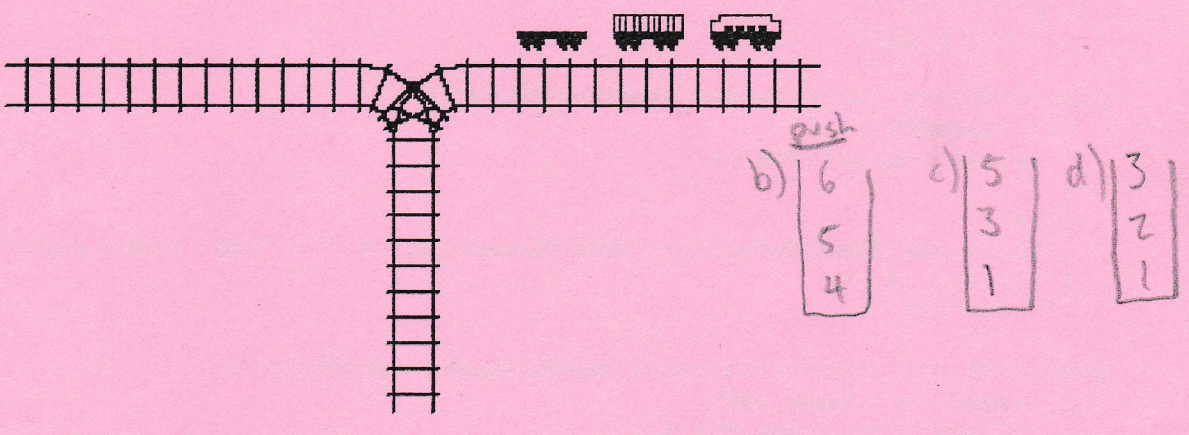


1. Suppose an empty stack is provided and an input stream contains the numbers 1 2 3 4 5 6, in this order. The numbers are read one by one from the input stream and either go directly into the output stream or are pushed onto the stack. When all numbers from the input stream are processed, all the numbers from the stack are popped one by one and sent to the same output stream. Determine whether each of the following represent valid or invalid output streams.

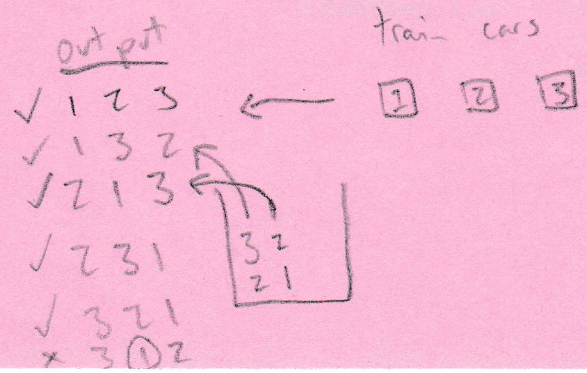
(a)	1	2	3	4	5	6
(b)	1	2	3	6	5	4
(c)	2	4	6	5	3	1
(d)	4	5	6	3	2	1
(e)	3	2	1	4	5	6



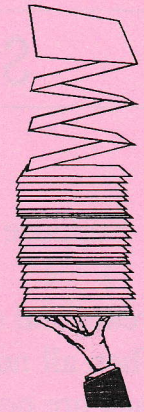
2.



Railroad cars numbered 1, 2, and 3 on the right track are to be permuted and moved along the left track. A car may be moved directly onto the left track, or it may be shunted onto the siding to be removed at a later time and placed on the left track. Find all possible combinations of cars that can be obtained on the left track by a sequence of these operations. For example, push 1, push 2, move 3, pop 2, pop1 arranges them in the order 3, 2, 1. Are any permutations not possible?



The Stack Class



```
public class Stack {
  private Node top;
  public Stack() {
    top = null;
  }  // end default constructor

  public boolean isEmpty() {
    return top ==  null;
  }  // end isEmpty

  public void push(Object newItem) {
    top = new Node(newItem, top);
  }  // end push

  public Object pop() throws StackException {
    if (!isEmpty()) {
      Node temp = top;
      top = top.getNext();
      return temp.getItem();
    }
    else {
      throw new StackException("StackException on " + "pop: stack empty");
    }  // end if
  }  // end pop

  public void popAll() {
    top = null;
  }  // end popAll

  public Object top() throws StackException {
    if (!isEmpty()) {
      return top.getItem();
    }
    else {
      throw new StackException("StackException on " + "top: stack empty");
    }  // end if
  }  // end top

  public Object clone() throws CloneNotSupportedException
  {
        Stack copy = new Stack();
        Node curr = top, prev = null;
        while (curr != null)
        {
                Node temp = new Node(curr.getItem());
                if (prev == null)
                        copy.top = temp;
                else
                        prev.setNext(temp);
                prev = temp;
                curr = curr.getNext();
        }
        return copy;
  }
}  // end Stack
```