Download the .pdf file and click on each problem to view code

Day-1 Problems (in detail) - 1st Problem + Extras

1. Reverse a Stack using Recursion. (See Video for reference)

Algorithm - First pop all elements of Stack one by one and hold them in a function 'Call Stack'.

2. Now insert one element, then to insert next element again pop this element and push that and then push the current element
3. Similarly to insert third element, pop the first two elements (storing them in Function Call Stack) and push the third element, now push these 2 elements back, similarly continue the same method until all elements are inserting

(inserting element at Bottom)

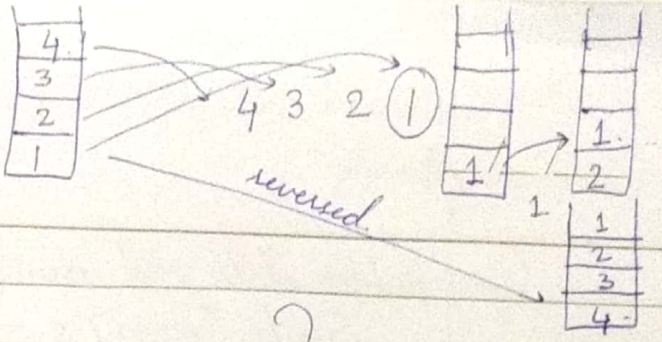
☆ In this code, we aren't implementing Stack from scratch, we are using the inbuilt function of Stack in java.

```
import java.util.Stack;
class Test {
    static Stack<Character> st = new Stack<>();   // can use Integer also (if giving int elements in stack)

static void insert_at_bottom(char x) {
        if(st.isEmpty()){
            st.push(x);
        else {
            char a = st.peek();
            st.pop();
            insert_at_bottom(x);
            st.push(a);
        }
    }
```

Step (2) and (3)

→




```
static void reverse() {
    if (st.size > 0) {
        char x = st.peek();
        st.pop();
        reverse();
        insert at bottom(x);
    }
}
```

step ① → step ② and ③

```
public static void main(String[] args) {
    st.push('1');
    st.push('2');
    st.push('3');
    st.push('4');
    System.out.println(st);
    reverse();
    System.out.println(st);
}
}
```

o/p

1 2 3 4

4 3 2 1

directly prints the stack (inbuilt)

Extra - Recursive approach to print a Stack manually. (From Bottom to Top)

```
import java.util.*;
class Extra {
static void PrintStack(Stack<Integer> s) {
    if (s.isEmpty())
        return;
    int x = s.peek();
    s.pop();
    PrintStack(s);
    System.out.Print(x+" ");
    s.push(x);
}
```

→

```
public static void main(String[]args){
    Stack<Integer> s = new Stack<Integer>();
      s.push(1);
      s.push(2);
      s.push(3);
      s.push(4);
      PrintStack(s);
    }
}
```

O/P :- 1 2 3 4

Extra 2:- Printing using anothes Stack. (Bottom to Top).

```
import java.util.*;
class Extra{
static void PrintStack(Stack<Integer> s){
    Stack<Integer> temp = new Stack<Integer>();
    while(s.isEmpty()==false){
        temp.push(s.peek());
        s.pop();
    }
    while(temp.isEmpty()==false){
        int t = temp.top();
        System.out.print(t+" ");
        temp.pop();
        s.push(t);
    }
}

psvm(){
    // main condition
}
```