Palindrome Using Recursion

```
import java.util.*;

import java.lang.*;

public class Palindrome_Using_Recursion {

        static int palindrome(int n , int temp)

        {   //when the enter number is zero it retrun temp

                if(n==0) {

                        return temp;

                }

                int r = n%10; // taking remainder

                temp = temp * 10 + r;

                return palindrome(n/10,temp);

        }

        public static void main(String[] args) {

                Scanner scan = new Scanner(System.in);

                int n = scan.nextInt();

                StringBuilder str = new StringBuilder();

                for(int i=0;i<=n;i++) {

                        str.append(scan.nextLine());

                }

                int number = Integer.parseInt(str.toString()); //Typescast to int

                int temp=0;

                int revnumber= palindrome(number, temp);

                System.out.println(number);  // print the actual number

                System.out.println(revnumber);  // print the reverse number

                if(number==revnumber) {

                        System.out.println("True");

                }

                else {

                        System.out.println("False");

          }
```

```
            }
}
```

Implement stack using Queue

```
import java.util.*;
public class Stack_Using_Queue {
            Queue<Integer> queue = new LinkedList<Integer>();

             void push(int val)
              {
                 //size of queue
                 int size = queue.size();

                 // add element
                 queue.add(val);

                 for (int i = 0; i < size; i++)
                 {

                    int x = queue.remove();
                    //add element to rear of queue
                    queue.add(x);
                 }
              }
            // removes the top element
              int pop()
              {
                 if (queue.isEmpty())
                 {
                    System.out.println("The stack is empty");
                    return 0;
            }
            int x = queue.remove();
            return x;
        }
      // top element of stack
        int top()
        {
            if (queue.isEmpty()) {
                return 0;
            }
            return queue.peek();
        }

        // true if Stack is empty else false
        boolean isEmpty()
        {
            return queue.isEmpty();
        }
        public static void main(String[] args) {
                Stack_Using_Queue stack = new Stack_Using_Queue();
                Scanner scan = new Scanner(System.in);
                int n = scan.nextInt();
                for(int i =0;i<n;i++) {
                        stack.push(scan.nextInt());
                }
                System.out.println(stack.top());
                stack.pop();
                System.out.println(stack.top());
                }

}
```