class Stack

{

    static final int MAX = 1000;

    int top;

    int a[] = new int[MAX]; // Maximum size of Stack

    boolean isEmpty()

    {

        return (top < 0);

    }

    Stack()

    {

        top = -1;

    }

    boolean push(int x)

    {

        if (top >= (MAX-1))

        {

            System.out.println("Stack Overflow");

            return false;

        }

        else

        {

            a[++top] = x;

            return true;

        }

    }

    int pop()

    {

        if (top < 0)

        {

            System.out.println("Stack Underflow");

            return 0;

        }

        else

        {

            int x = a[top--];

            return x;

        }

    }

}

Stack<Type> s = new Stack();

\*\* s.pop().equals(5) not s.pop() == 5

**Determine whether an integer is a palindrome.**

class Solution {

public boolean isPalindrome(int x) {

if (x < 0)

return false;

int midpoint = calcMidpoint(x, 0);

int n = x;

Stack<Integer> s = new Stack();

while (midpoint != 0) {

s.push(n%10);

n /= 10;

midpoint--;

}

// step over middle number

if (isOddLength(x, 0))

n /= 10;

while (!s.isEmpty()) {

if (!s.pop().equals(n%10))

return false;

n /= 10;

}

return true;

}

public int calcMidpoint(int n, int midpoint) {

while (n != 0) {

midpoint++;

n /= 10;

}

return midpoint / 2;

}

public boolean isOddLength(int n, int length) {

while (n != 0) {

length++;

n /= 10;

}

if (length % 2 == 1)

return true;

return false;

}

}