1. Power of a number

Code:

package Assignment1;

public class PowerOfNumber {

    static void method1(int num,int pow,long ans){

        if(pow==0){

            System.out.println("By stack building method: "+ans);

            return;

        }

        method1(num,pow-1,ans\*num);

    }

    static long method2(int num,int pow){

        if(pow==0)

            return 1;

        return num\*method2(num,pow-1);

    }

    public static void main(String[] args) {

        method1(2,3,1);

        System.out.println("By stack falling method: "+method2(2, 3));

    }

}

1. Count zero in given number

Code:

package Assignment1;

public class CountZeros {

    public static void method1(int num,int zeros){

        if(num==0){

            System.out.println("Using stack building method: "+zeros);

            return;

        }

        int dig = num%10;

        if(dig==0)

            zeros++;

        method1(num/10,zeros);

    }

    public static int method2(int num){

        if(num==0)

            return 0;

        return ((num%10)==0?1:0) + method2(num/10);

    }

    public static void main(String[] args) {

        method1(1000100,0);

        System.out.println("Using stack falling method: "+method2(1000100));

    }

}

1. Sum of N natural numbers

Code:

package Assignment1;

public class SumOfNNaturalNumbers {

    public static void method1(int n,int sum){

        if(n==0){

            System.out.println("Using Stack Building Method: "+sum);

            return;

        }

        method1(n-1,sum+n);

    }

    public static int method2(int n){

        if(n==0)

            return 0;

        return n+method2(n-1);

    }

    public static void main(String[] args) {

        method1(10,0);

        System.out.println("Using Stack Falling Method: "+method2(10));

    }

}

1. 1/1^1 + 2/2^2 + 3/3^3 + … + n/n^n

Code:

package Assignment1;

// Pattern is 1/1^1 + 2/2^2 + ..... + n/n^n

public class SumOfPattern {

    public static void method1(int n,double ans){

        if(n==0){

            System.out.println("Using stack building method: "+ans);

            return;

        }

        ans+=n/Math.pow(n,n);

        method1(n-1, ans);

    }

    public static double method2(int n){

        if(n==0)

            return 0;

        double val = n/Math.pow(n, n);

        return val+method2(n-1);

    }

    public static void main(String[] args) {

        method1(5, 0);

        System.out.println("Using stack falling method: "+method2(5));

    }

}

1. Check given number is prime or not

Code:

package Assignment1;

public class CheckPrime {

    public static void method1(int num,int div){

        if(div==(num/2)+1){

            System.out.println("By stack building method: "+true);

            return;

        }

        else if(num%div==0){

            System.out.println("By stack building method: "+false);

            return;

        }

        method1(num,div+1);

    }

    public static boolean method2(int num,int div){

        if(div==(num/2)+1)

            return true;

        return (num%div!=0) && method2(num,div+1);

    }

    public static void main(String[] args) {

        method1(14,2);

        System.out.println("Using stack falling method: "+method2(13, 2));

    }

}

1. Check given number is Armstrong or not

Code:

package Assignment1;

public class CheckArmstrong {

    public static int findDig(int num){

        if(num==0)

            return 0;

        return 1+findDig(num/10);

    }

    public static void method1(int origNum, int num,int val,int dig){

        if(num==0){

            System.out.println("Using stack building method: "+(origNum==val));

            return;

        }

        method1(origNum,num/10,val+(int)Math.pow(num%10,dig),dig);

    }

    public static int method2(int num,int dig){

        if(num==0)

            return 0;

        return (int)Math.pow(num%10,dig) + method2(num/10,dig);

    }

    public static void main(String[] args) {

        int dig = findDig(153);

        method1(153, 153, 0, dig);

        System.out.println("Using stack falling method: "+(154==method2(154,dig)));

    }

}

1. Check Palindrome String

Code:

package Assignment1;

public class CheckPalindrome {

    public static void method1(String s, int ptr){

        if(ptr>=s.length()/2){

            System.out.println("Using stack building method: "+true);

            return;

        }

        else if(s.charAt(ptr)!=s.charAt(s.length()-ptr-1)){

            System.out.println("Using stack building method: "+false);

            return;

        }

        method1(s,ptr+1);

    }

    public static boolean method2(String s,int ptr){

        if(ptr>=s.length()/2)

            return true;

        return (s.charAt(ptr)==s.charAt(s.length()-ptr-1)) && method2(s, ptr+1);

    }

    public static void main(String[] args) {

        method1("naman", 0);

        System.out.println("Using stack falling method: "+method2("naman", 0));

    }

}

1. Last Index of a particular number

Code:

package Assignment1;

public class LastIndexOf {

    public static void method1(int []arr,int val,int ptr){

        if(ptr==-1){

            System.out.println("Using stack building method position: " +ptr);

            return;

        }

        else if(arr[ptr]==val){

            System.out.println("Using stack building method position: "+ptr);

            return;

        }

        else

            method1(arr, val, ptr-1);

    }

    public static int method2(int arr[],int val,int ptr){

        if(ptr==-1)

            return ptr;

        else if(arr[ptr]==val)

            return ptr;

        return method2(arr, val, ptr-1);

    }

    public static void main(String[] args) {

        method1(new int[]{1,2,3,4,4,4,7},4,6);

        System.out.println("Using stack falling method position: "+method2(new int[]{1,2,3,4,4,4,7},4,6));

    }

}

1. Count occurrences of number

Code:

package Assignment1;

public class CountOccurances {

    public static void method1(int []arr,int val,int ptr,int occ){

        if(ptr==arr.length){

            System.out.println("Using stack building method occurances are: "+occ);

            return;

        }

        method1(arr,val,ptr+1,(arr[ptr]==val)?occ+1:occ);

    }

    public static int method2(int []arr,int val,int ptr){

        if(ptr==arr.length)

            return 0;

        return ((arr[ptr]==val)?1:0) + method2(arr,val,ptr+1);

    }

    public static void main(String[] args) {

        method1(new int[]{1,2,3,4,4,4,5,4,4,7,8},4,0,0);

        System.out.println("Using stack falling method occuracnes are: "+method2(new int[]{1,2,3,4,4,4,5,4,4,7,8},4,0));

    }

}

1. Store the get all found element index in an array

Code:

package Assignment1;

public class StoreIndicesOfFoundElement {

    public static void printArray(int []arr){

        for(int i=0;i<arr.length;i++)

            System.out.print(arr[i]+" ");

    }

    public static int[] method1(int arr[],int ptr,int val,int occ){

        if(ptr == arr.length)

            return new int[occ];

        if(arr[ptr]==val)

            occ++;

        int curr[] = method1(arr,ptr+1,val,occ);

        if(arr[ptr]==val)

            curr[occ-1] = ptr;

        return curr;

    }

    public static void main(String[] args) {

        System.out.println("The position of occurances are: ");

        printArray(method1(new int[]{1,2,2,1,1,3,1,1,3,1,2}, 0, 1, 0));

    }

}

1. Replace all occurrences.

Code:

package Assignment1;

public class SearchAndReplace {

    public static void printArray(int []arr){

        for(int i=0;i<arr.length;i++)

            System.out.print(arr[i]+" ");

    }

    public static void method1(int[]arr,int val,int repVal,int ptr){

        if(ptr==arr.length){

            System.out.println("Using stack building method the array will be: ");

            printArray(arr);

            return;

        }

        else if(arr[ptr]==val)

            arr[ptr]=repVal;

        method1(arr, val, repVal, ptr+1);

    }

    public static int[] method2(int[]arr,int val,int repVal,int ptr){

        if(ptr==arr.length){

            return new int[ptr];

        }

        int curr[] = method2(arr,val,repVal,ptr+1);

        if(arr[ptr]==val)

            curr[ptr] = repVal;

        else

            curr[ptr] = arr[ptr];

        return curr;

    }

    public static void main(String[] args) {

        method1(new int[]{1,1,1,2,3,4,1,1,5,1,6},1,9,0);

        System.out.println("\nBy stack falling method the array will be: ");

        printArray(method2(new int[]{1,1,1,2,3,4,1,1,5,1,6},1,9,0));

    }

}

12. Given a String find the adjacent element if it is same as the previous element so replace the element with #.

e.g "aabbcc"

output a#b#c#

public class test {

    public static void main(String[] args) {

        String s = "aaabbddcc";

        changes(s, "");

    }

    public static void changes(String s, String st) {

        if (s.length() <= 1) {

            System.out.println(st);

            return;

        }

        char ch = s.charAt(0);

        char ch1 = s.charAt(1);

        if (ch == ch1) {

            st = st + ch + "\*";

            changes(s.substring(2), st);

        } else {

            st = st + ch;

            changes(s.substring(1), st);

        }

    }

13. Given a String find the adjacent element if it is same as the previous element so add the star in between

e.g "aabbcc"

output a\*ab\*bc\*c

public class test {

    public static void main(String[] args) {

        String s = "aaabbddcc";

        changes(s, "");

    }

    public static void changes(String s, String st) {

        if (s.length() <= 1) {

            System.out.println(st);

            return;

        }

        char ch = s.charAt(0);

        char ch1 = s.charAt(1);

        if (ch == ch1) {

            st = st + ch + "\*";

            changes(s.substring(2), st);

        } else {

            st = st + ch;

            changes(s.substring(1), st);

        }

    }

}