Recursion

Print Desending order from n to 1 using recursion

```
public class recursions {
    public static void dec(int n){
        if(n==1){// base case
            System.out.println(1);
            return;
        }
        System.out.println(n+" "); // print n
        dec(n-1);// decrease
    }
    public static void main(String args[]){
        int n=5;
        dec(n);
    }

}
Output:5 4 3 2 1
```

Print ascending order from 1 to n using recursion

```
public class recursions {
    public static void dec(int n){
        if(n==1){// base case
            System.out.println(1);
            return;
        }

        dec(n-1);// decrease
        System.out.println(n);

    }
    public static void main(String args[]){
        int n=5;
        dec(n);
    }

}
Output: 1 2 3 4 5
```

Print factorial using recursion

```
public class recursions {
    public static int fact(int n){
        if(n==1){// base case
           return 1;
        }
         int fnm1=fact(n-1);
         int factn=n*fnm1;
         return factn;


    }
    public static void main(String args[]){
        int n=5;
      System.out.println(fact(n));
    }

}
Output: 120
```

Sum of numbers till n :

```
public class recursions {
    public static int fact(int n){
        if(n==1){// base case
           return 1;
        }
         int fnm1=fact(n-1);
         int factn=n+fnm1;
         return factn;


    }
    public static void main(String args[]){
        int n=5;
      System.out.println(fact(n));
    }

}
Output: 15
```

Fibonacci series:

```
public class recursions {
    public static int fabi(int n){
        if(n==0|| n==1 ){// base case
           return n;
        }
         int fnm1=fabi(n-1);
         int fnm2=fabi(n-2);
        int fabn=fabi(n-1)+fabi(n-2);

        return fabn;


    }
    public static void main(String args[]){
        int n=5;
      System.out.println(fabi(n));
    }

}
Output: 5
```

Check if a given array is sorted or not:

```
public class recursions {
    public static boolean issorted(int arr[], int i){
        if(i ==arr.length-1 ){// base case
           return true;
        }
        if(arr[i]>arr[i+1]){
            return false;
        }
        return issorted(arr,i+1);

    }
    public static void main(String args[]){
        int arr[]={1,2,3,4,5};
      System.out.println(issorted(arr, 0));
    }

}
Output: true
```

Find the first occurrence of an element in a array

```
public class firstoccurence {
    public static int FOcc(int arr[],int key,int i){

        // base case
        if(i==arr.length){
            return -1;
        }
        if(arr[i]==key){
            return i;
        }
        return FOcc(arr,key ,i+1);
    }
    public static void main(String args[]){
        int arr[]={4,3,2,1,5,6,7,8};
       System.out.println( FOcc(arr,5,0));
    }

}
Output:4
```

Find the last occurrence of an element in a array

```
public class lastoccurence {

    public static int LOcc(int arr[], int key, int i){
//base case
    if(i==arr.length){
        return -1;
    }
    int isfound=LOcc(arr, key, i+1);
    if(isfound==-1&&arr[i]==key){
        return i;
    }
    return isfound;

    }
    public static void main(String args[]){
        int arr[]={1,2,3,4,5,6,7,8,5};
        System.out.println(LOcc(arr, 5, 0));
    }
}
Output: 8
```

Print x^n

```
public class printxpowern {
    public static int power(int x,int n){
        //base case
        if(n==0){
            return 1;
        }
        int nm1=power(x,n-1);
        int pn=x*nm1;
        return pn;
        //return x*pow(x,n-1)//single line code

    }
    public static void main(String args[]){
        System.out.println(power(2, 5));
    }

}
Output: 32
```

Optimized code

```
 public class printxpowern {
  public static int optimalpow(int x,int n){
    if(n==0){
        return 1;
    }
    int halfpow=optimalpow(x, n/2);
    int optimalpow=halfpow*halfpow;
    if(n%2!=0){
        optimalpow=x*optimalpow;
    }
    return optimalpow;

    }
    public static void main(String args[]){
        int x=2;
        int n=5;
        System.out.println(optimalpow(x, n));
    }
```

```
}
Output: 32
```

*Tiling problem (amazon)

```
public class TILINGPROBLEM {
    public static int tilingprob(int n){
        // base case
        if(n==0 || n==1){
            return 1;
        }
        //work what to do
        //vertical -> f(n-1)
        int fnm1vertical=tilingprob(n-1);

        // horizontal _> f(n-2)
        int fnm2horizontal =tilingprob(n-2);

        //total ways
        int totways= fnm1vertical + fnm2horizontal;
        return totways;
      //return tilingprob(n-1)+tilingprob(n-2);

    }
    public static void main(String args[]){
        System.out.println(tilingprob(4));
    }

}
Output: 5
```

WAF to remove duplicates in a String(Amazon,google)

```
public class removeduplicatesinstring {
    public static void removeduplicate(String str,int idx,StringBuilder
sb,boolean map[]){
        // base case
        if(idx==str.length()){
            System.out.println(sb);
            return;
```

```
        }
        //work to do
        //compare current element
        char currchar=str.charAt(idx);

        if(map[currchar-'a']==true){
            removeduplicate(str, idx+1, sb, map);

        }
        else{
         map[currchar -'a'] = true;
            removeduplicate(str, idx+1, sb.append(currchar), map);
        }
    }
    public static void main(String args[]){
        String str="appnnacoolleggge";
        removeduplicate(str, 0, new StringBuilder(""), new boolean[26]);

    }

}
Output:
apncoleg
```

Friends pairing problem:

```
public class friendspairing {
    public static int friendspairing(int n){
        // base case
        if(n==1||n==2){
            return n;
        }
        //work
        // // single
        // int fnm1=friendspairing(n-1);
        // //pairs
        // int fnm2=friendspairing(n-2);
        // int pairs=(n-1)*fnm2;
        // int totpairs=fnm1+pairs;
        // return totpairs;
        return friendspairing(n-1)+(n-1)*friendspairing(n-2);
    }
```

```
    public static void main(String args[]){
        System.out.println(friendspairing(3));
    }

}
Output: 4
```

Binary numbers in a of a string without consecutive ones(paytm)

```
public class binarystringproblem {
    public static void binarystring(int n,int lastplace, String str){
        //base case
        if(n==0){
            System.out.println(str);
            return;
        }
        //work
        if(lastplace==0){
            binarystring(n-1, 0, str+"0");
            binarystring(n-1, 1, str+"1");
        }
        else{
            binarystring(n-1, 0, str+"0");
        }
       //or
        // binarystring(n-1, 0, str+"0");
        // if(lastplace==0){
        //     binarystring(n-1, 1, str+"1");

        // }




    }
    public static void main(String args[]){
        binarystring(3, 0,  new String(""));
    }

}
```

```
Output:
000
001
010
100
101
```

Print the occurences(indexes) of an element in array

```
public class printindexofarray {
    public static void  printindex(int arr[],int key ,int index){
        //base case
        if(index==arr.length){
            return ;
        }
        //work
        if(arr[index]==key){
           System.out.print(index+" ");

        }

            printindex(arr, key, index+1);


    }
    public static void main(String args[]){
        int arr[]= {3,2,4,5,6,2,7,2};
        printindex(arr, 2, 0);
    }

}
Output:
157
```

Convert number into string

```
public class convertnumbertostring {
```

```
    //create string
    static String
digits[]={"zero","one","two","three","four","five","six","seven","eight","nine"};

    public static void printstring(int nbr){
        //base case
        if(nbr==0){
            return;
        }
        int lastdigit=nbr%10;// pick lastdigit
        printstring(nbr/10);//update nmbr
        //print number
        System.out.print(digits[lastdigit]+ " ");
    }
    public static void main(String args[]){
        printstring(2002);
    }

}
Output:
Two zero zero two
```

Find the length of string

```
public class findstringlength {
    public static int length(String str, int i){
        char c=str.charAt(i);
        if(c == ' ') {
            return i;

        }
        return length(str, i+1);
        //return length(str.substring(1)) +1;

}
public static void main(String args[]){
    String str="abcde ";
    System.out.println(length(str,0));
}
}
```

Method 2:

```
public class findstringlength {
```

```
    public static int length(String str){
        if(str.length()==0){
            return 0;

        }
        return length(str.substring(1)) +1;

}
public static void main(String args[]){
    System.out.println(length("abcde"));
}
}
```

Find the number of substrings with starting and ending with same alphabet

```
public class findingsubstrings {
    public static int printsubstring(String str,int i,int j,int n){
        if(n<=0){
            return 0;
        }
        if(n==1){
            return 1;
        }
    int subs=printsubstring(str, i+1, j, n-1)+ printsubstring(str, i, j-1, n-1)-
printsubstring(str, i+1, j-1, n-2);
 if(str.charAt(i)==str.charAt(j)){
    subs++;
 }
return subs;
    }
    public static void main(String args[]){
        String str="aba";
        int n=str.length();
       System.out.println( printsubstring(str, 0, n-1, n));

    }

}
Output; 4
```

Tower of Hanoi

```
public class towerofhanoi {

    public static void towerofhanoi(int n, String src, String helper,String
dest){
        //base case
        if(n==1){
         System.out.println(" transfered disk"+ n+"from"+ src+"to"+dest);
         return;
        }
        towerofhanoi(n-1, src, dest, helper);
        System.out.println(" transfered disk"+ n+"from"+ src+"to"+dest);
        towerofhanoi(n-1, helper, src, dest);
    }
    public static void main(String args[]){
        int n=3;
        towerofhanoi(n, "S", "H", "D");
    }

}
Output:
transfered disk 1fromStoD
 transfered disk 2fromStoH
 transfered disk  1fromDtoH
 transfered disk 3fromStoD
 transfered disk 1fromHtoS
 transfered disk 2fromHtoD
 transfered disk 1fromStoD
```

reverse a string using recursion

```
public class reverseastring {
    public static void reversestring(String str, int idx){
        //base case
        if(idx==0){
            System.out.println(str.charAt(idx));
            return;
        }
        System.out.print(str.charAt(idx));
        reversestring(str, idx-1);
    }
```

```
    public static void main(String args[]){
        String str="saikiran";
        reversestring(str, str.length()-1);
    }

}
Output:
narikias
```