Day 3 Grid Unique Paths

```
class gridunique{
        static int griduniquePaths(int m ,int n ){
                int N = n+m-2;
                int r = m-1;
                double res=1;
                for(int i=1;i<=r;i++){
                        res = res * (N-r+ i)/ i;
                }
                return (int) res;
        }
        public static void main(String[] args) {
                int m=3 ,n=2;
                System.out.println("Total unique path is :" + 
griduniquePaths(m,n));

        }
}
```

// Day 3.3 Majority Element

```
import java.util.*;
class majorityelem{
    // most optimized approach using moore's method
    static int majority(int[] nums){
        int cnt =0;
        int candidate= 0;
        for (int num  : nums ) {

            if(cnt == 0) candidate = num;

            if(num == candidate){
                cnt+=1;
            }
            else
            {
                cnt-=1;
            }
        }
        return candidate;
    }
    // Its time Complexity is O(Nlogn)
```

```
          static int majority1(int[] nums){


       HashMap<Integer,Integer> hs = new HashMap<Integer,Integer>();
       for (int i  : nums) {
           if(hs.containsKey(i)){
                    int cnt = hs.get(i);
                    //System.out.println(cnt);
                    if(cnt+1 >= nums.length/2) return i;
                    hs.replace(i, cnt+1);
           }
           else{
                    hs.put(i,1);
           }


       }
                     return 0;
          }
          public static void main(String[] args) {
           int nums[]= {4,4,3,3,4,4,4,4};
            System.out.println("The Majority element is  :"+
majority1(nums));
          }}
```

```
// day3 Majority element II

import java.util.*;
class majorityelem2{
        static List<Integer> majorityelem2_fn(int a[],int n){
                int num1 =-1,num2 =-1, el1=0, el2=0, ct1=0,ct2=0;

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

                        if(a[i] == num1){
                                ct1++;
                        }
                        else if(a[i]== num2){
                                ct2++;
                        }
                        else if (ct1==0){
                                num1 = a[i];
                                ct1=1;
                        }
                        else if(ct2 ==0){
                                num2 = a[i];
                                ct2 =1;
```

```
          }
          else{
              ct1--;
              ct2--;
          }
      }

      List<Integer> ls = new ArrayList<>();
ct1 =0;
ct2=0;
for( int i: a){
 if(i == num1){
     ct1++;
 }
 else if(i == num2){
     ct2++;
 }
}
      if( ct1 > n/3 ){
          ls.add(num1);
      }
      if(ct2 > n/3){
        ls.add(num2);
    }

    return ls;

}
static List<Integer> majorityelem2_fn1(int a[],int n){

    HashMap<Integer,Integer> hs = new HashMap<>();
    List<Integer> ls =new ArrayList<>();
    int n1= n/3;
    for ( Integer i : a) {
        if (hs.containsKey(i)) {
            int cnt= hs.get(i);
            if(cnt+1> n1)
                ls.add(i);
            hs.replace(i,cnt+1);
        }
        else{
            hs.put(i,1);
        }
    }
```

```
          return ls;
     }
     public static void main(String[] args) {
          int a[]={1,1,1,3,3,2,2,2};
          int n = a.length;
          System.out.println(majorityelem2_fn1(a,n));

     }
}

// Merge sort
// its time complexity as usual O(nlogn) space complexity : O(n)
class merge{
  static void merge1(int arr[],int l, int mid ,int h){
     int n1 = mid-l+1;
     int n2=h-mid;
     int a[] = new int[n1];
     int b[] = new int[n2];
    for (int i=0;i< n1 ;i++ ) {
     a[i]= arr[l+i];
    }
    for (int i=0;i< n2 ;i++ ) {
       b[i]= arr[mid+1+i];
      }

       int j=0,i=0,k=l;
       while(i <n1&& j<n2 ){
            if(a[i] < b[j]){
                 arr[k++]=a[i++];
            }
            else{
                 arr[k++]=b[j++];
            }
       }

       while(i<n1){
            arr[k++]=a[i++];
       }
       while(j<n2){
            arr[k++]=b[j++];
       }
    }
       static void mergefun(int arr[],int l,int h){
      if(l<h){
```

```
      int mid=(l+h)/2;
      mergefun(arr, l ,mid);
      mergefun(arr, mid+1 ,h);
      merge1(arr,l,mid,h);
    }

     }
     public static void main(String[] args) {
          int arr[]= { 5,4,3,2,1};
          mergefun(arr,0,arr.length-1);
          System.out.println("The Array is :");
    for(int i=0;i<=arr.length-1;i++){
     System.out.print(arr[i]+" ");
    }
     }
}
```

```
//Day 3 pow(x,n)
import java.util.*;
class powxn{
   static double powxn_fun(double x, int n){
      double ans = 1.0;
      long nn = n;
      if(nn > 0) nn = -1 * nn;
      while(nn>0){
          if (nn % 2 == 1) {
              ans = ans * x;
              nn=nn-1;
          }
          else{
              x = x * x ;
              nn=nn/2;
          }
      }
      if(n < 0 ) ans=(double)(1.0)/(double)ans;
      return ans;
  }
  public static void main(String[] args) {

double x= 2.0;
int n = 3;
  //    Scanner sc=new Scanner(System.in);
      // double x=sc.nextDouble();
```

```
        // int n=sc.nextInt();
        System.out.println("The Answer is :" +  powxn_fun(x,n));


    }
}



// Day 3.6 Reverse a pair



import java.util.*;


public class reverse_pair{


static  private int merge(int a[],int lo,int mid ,int hi){


   int nums[]=new int[hi-lo+1];
        int p=lo, q=mid+1,cnt=0,index=0;
        while(p<=lo && q <= hi){
            if((long)a[p] >  2* (long) a[q]){
                cnt+= mid-p +1;
                q++;
        }
        else{
                p++;
        }
}
p=lo;
q=mid+1;
while(p<=mid && q <= hi){
        if (a[p] < a[q]) {
                nums[index++]=a[p++];
        }
        else{
           nums[index++]=a[q++];
        }
}

// for remaining elements


while(p<=mid){
        nums[index++]=a[p++];
}
while(q<=hi){
```

```
            nums[index++]=a[q++];
        }
        System.arraycopy(nums, 0 ,a ,lo, (hi-lo)+1);
        return cnt;
    }



static int mergesort(int a[],int lo,int hi){
    if(lo>=hi) return 0;
    int count =0;
    int mid = lo + (hi-lo)/2;
    count+=mergesort(a,lo,mid);
    count+=mergesort(a,mid+1,hi);
    count+=merge(a,lo,mid,hi);
    return count;
}
static int reverse_P(int arr[]){
    int n=arr.length;
    return mergesort(arr,0,n-1);
}
    public static void main(String[] args) {
        int arr[] ={2,4,3,5,1};
```

```
          int pair = reverse_P(arr);
          System.out.println("The reverse pair is  :" +  pair);
     }
}
// Day 3.2  Search Array in 2D matrix
import java.util.*;
class search2d{

     static boolean search2darray(int matrix[][], int target){
    if(matrix.length == 0) return false;
          int n = matrix.length;
          int m= matrix[0].length;
          int lo=0;
          int hi=(n*m)-1;
          while(lo<= hi){
            int mid = (lo + (hi-lo) / 2);
               if(matrix[mid/m][mid%m]== target){
                    System.out.println(" The element is present at 
index matrix["+(mid/m) +"]"+ "["+(mid%m)+"]");
                    return true;
               }
               if (matrix[mid/m][mid%m] < target) {
                lo = mid+1;
            }
            else{
                hi= mid -1;
            }
        }
  return false;
    }
    public static void main(String[] args) {
        int arr[][]={{1,3,6},{10,19,20},{22,30,55}};
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the target element you want to search :");
        int target = sc.nextInt();
        System.out.println("The target element is Present :"+ search2darray(arr,target));
    }
}
```