**Data Structure in (Java)**

**Topic:** Array (Basic Questions of arrays)

Question 1: Java Program to find largest two number in array.

Solution:

Bubble sorting used for solving this particular problem.

import java.util.Scanner;

public class Array {

public static void main(String[] args) {

Scanner s =new Scanner(System.in);

System.out.println("Enter a size of array");

int n=s.nextInt();

int []arr=new int[n];

System.out.println("Enter a element of array");

for(int i=0;i<arr.length;i++)

{

arr[i]=s.nextInt();

}

System.out.println("sort the given array");

for(int i=0;i<arr.length;i++)

{

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

}

for(int i=0;i<arr.length;i++)

{

int temp=0;

for(int j=i+1;j<arr.length;j++)

{

if(arr[j]<arr[i])

{

temp=arr[j];

arr[j]=arr[i];

arr[i]=temp;

}

}

}

System.out.println("After sorting of array:");

for(int i=0;i<arr.length;i++)

{

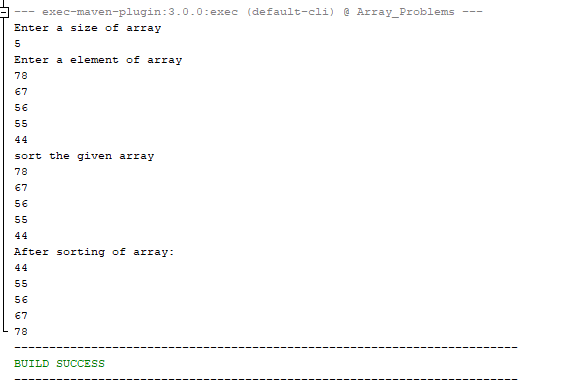
System.out.println(arr[i]+" ");

}

}

}

Output:



Question 1: Java Program to find Second largest number and smallest element in array.

Solution:

class second\_largest\_element

{

public void find\_large\_element(int [] arr, int length)

{

int large=arr[0];

int second\_large=arr[1];

int temp;

int max=arr[0];

int min=arr[0];

for(int i=0;i<length;i++)

{

if(max<arr[i])

{

max=arr[i];

}

if(min>arr[i])

{

min=arr[i];

}

}

System.out.println("largest element:"+max);

System.out.println("minimum element:"+min);

int sec\_max=arr[0];

for(int i=1;i<length;i++)

{

if(arr[i]<max&&arr[i]>sec\_max)

{

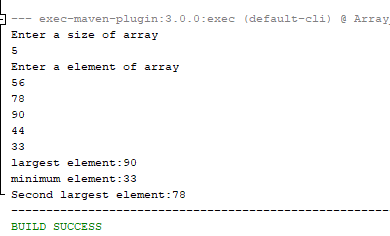
sec\_max=arr[i];

}

}

System.out.println("Second largest element:"+sec\_max);

}

}.

Question: find the minimum sum of array

Solution: In this program I used the karden algorithm:

class Minimum

{

public void find\_min\_sum(int [] arr,int length)

{

int sum1=0;

int sum2=arr[0];

for(int i=0;i<length;i++)

{

sum1=0;

for(int j=i+1;j<length;j++)

{

sum1+=arr[j];

if(sum2>sum1)

{

sum2=sum1;

}

}

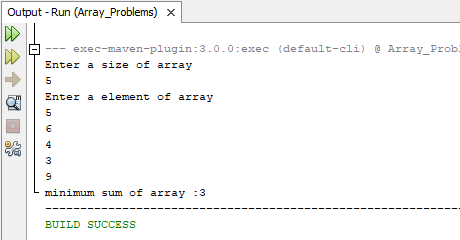
}

System.out.println("minimum sum of array :"+sum2);

}

}

Output:



Question: find the maximum sum of continuous subarray.

Solution:

class Maximum

{

public void find\_max\_sum(int [] arr,int length)

{

int sum1=0;

int max=arr[0];

for(int i=0;i<length;i++)

{

sum1+=arr[i];

if(max<sum1)

{

max=sum1;

}

if(sum1<0)

sum1=0;

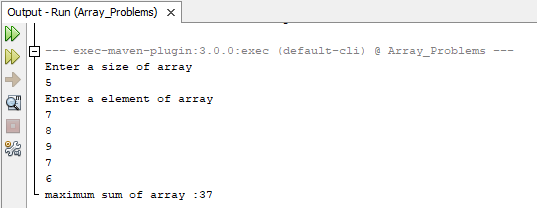
}

System.out.println("maximum sum of array :"+max);

}

}

Output:



Question: Find the even and odd no. of array and print.

Solution:

class Even\_Odd

{

public void find\_even\_odd(int []arr,int length)

{

int []b=new int[length];

int []c=new int[length];

int j=0,k=0;

for(int i=0;i<length;i++)

{

if(arr[i]%2==0)

{

b[j]=arr[i];

j++;

}

else

{

c[k]=arr[i];

k++;

}

}

System.out.println("Even no. of array:");

for(int i=0;i<j;i++)

{

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

}

System.out.println("\n");

System.out.println("odd no. of array:");

for(int i=0;i<k;i++)

{

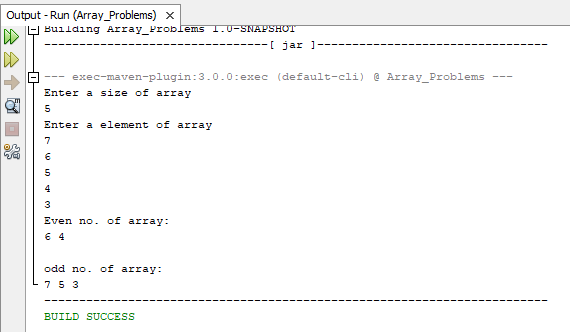
System.out.print(c[i]+" ");

}

}

}

Output:



Question: Insert an element in array:

Solution:

In this problem array capacity,capacity+1, so array created greate than 1 of its size and we add one value shift the values till the position user wants:

I=(n-1)// for loop condition,

public class Array {

public static void main(String[] args) {

Scanner s=new Scanner(System.in);

System.out.println("ENter a size of an array");

int n=s.nextInt();

System.out.println("ENter a Element of an array");

int [] arr=new int[n+1];

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

{

arr[i]=s.nextInt();

}

System.out.println("ENter a element and position your want to insert an element ");

int value= s.nextInt();

int pos=s.nextInt();

for(int i=(n-1);i>=(pos-1);i--)

{

arr[i+1]=arr[i];

}

arr[pos-1]=value;

System.out.println("After inserting an element of array");

for(int i=0;i<arr.length;i++)

{

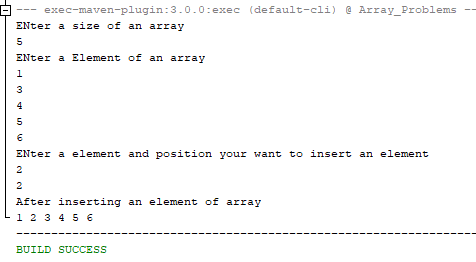
System.out.print(arr[i]+" ");

}

}

}

Output:



Question: Delete an element of an array.

Solution:

public void delete\_element(int[]arr,int length)

{

Scanner s=new Scanner(System.in);

System.out.println("Enter a element your want to delete");

int delete=s.nextInt();

int pos=0;

for(int i=0;i<arr.length;i++)

{

if(arr[i]==delete)

pos=i;

}

for(int i=pos;i<arr.length-1;i++)

{

arr[i]=arr[i+1];

}

System.out.println("After deletion of array element");

for(int i=0;i<arr.length-2;i++)

{

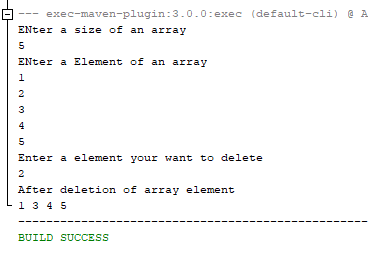
System.out.print(arr[i]+" ");

}

}

}

Output:



Question: Split the array.

Solution:

class Split\_Array{

public void split\_array(int [] a,int length)

{

Scanner s=new Scanner(System.in);

System.out.println("Enter a position you want to split the array ");

int pos=s.nextInt();

int [] b=new int[length];

int []c=new int[length];

int k=0,x=0,i;

for(i=0;i<pos;i++)

{

b[k]=a[i];

k++;

}

for(int j=i;j<length;j++)

{

c[x]=a[j];

x++;

}

System.out.println("After spliting the array ");

for(int l=0;l<k;l++)

{

System.out.print(b[l]+" ");

}

System.out.println("\n");

for(int m=0;m<x;m++)

{

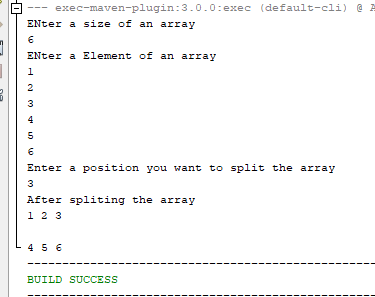
System.out.print(c[m]+" ");

}

}

}

Output:



Question: Moves the negative element in right side and positive on left side.

Solution:

class Moves\_Negative\_Element{

public void moves\_element(int []a, int length)

{

int rigth=0;

int left=length-1;

int temp=0;

while(rigth<=left)

{

if(a[rigth]<0 && a[left]<0)

{

rigth++;

}

else if(a[rigth]>0 && a[left]<0)

{

temp=a[rigth];

a[rigth]=a[left];

a[left]=temp;

left--;

rigth++;

}

else if(a[rigth]>0 && a[left]>0)

{

left--;

}

else

{

rigth++;

left--;

}

}

System.out.println("After a moves all negative element of array on rigth!!");

for(int i=0;i<a.length;i++)

{

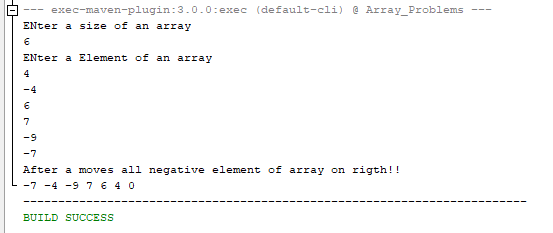
System.out.print(a[i]+" ");

}

}

}

Output:



Question: Sorting 0,1,and 2 without using any sorting algorithm.

Solution:

Approach 1: sort the array.

Approach 2: count the 0,1and 2.

Approach 3: Dutch national flag algorithm.

class sort\_0\_1\_2{

public void sort\_z\_o\_t(int []arr,int length)

{

int low=0;

int mid=0;

int high=length-1;

int temp=0;

while(mid<=high)

{

if(arr[mid]==0)

{ temp=arr[low];

arr[low]=arr[mid];

arr[mid]=temp;

low++;

mid++;

}

else if(arr[mid]==1)

{

mid++;

}

else

{

temp=arr[high];

arr[high]=arr[mid];

arr[mid]=temp;

high--;

}

}

System.out.print("After sorting 0 1 and 2 \n");

for(int i=0;i<length;i++)

{

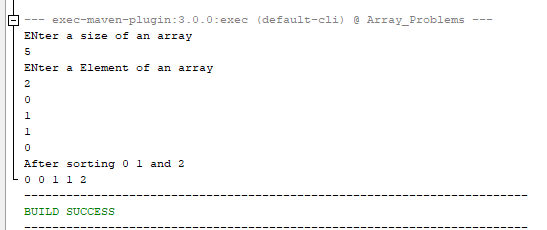
System.out.print(arr[i]+" ");

}

}

}

Output:



Question: Cyclic rotation of an array.

Solution:

class Cyclic\_Rotation{

public void rotation\_of\_array(int [] arr1,int length)

{

int temp=arr1[length-1];

for(int i=arr1.length-1;i>0;i--)

{

arr1[i]=arr1[i-1];

}

arr1[0]=temp;

System.out.print("After rotation of array1 ");

for(int i=0;i<arr1.length;i++)

{

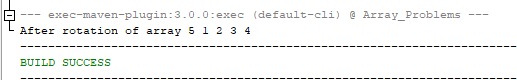
System.out.print(arr1[i]+" ");

}

}

}

Output:



Question: Minimize the different between the height.

Solution:

In this problem we minimize the Diffirence of max value and min value. Here we have the Kth value. We can add or sub the kth value from our arrays element and minimize the difference.

Step1: Sort the array.

Step2: 1.check if(a[i]>=k)

Then,

Find min and max.

2. we can add the kth value in the smallest value and subtract the kth value from the max, find the max.

3. and a[0]+k,a[i]-k,find the min.

Else return the -1,

Step 3. Find the result=max-min;

Step 4. Retrun result.

class Minimize\_Diffirence{

public int getMinDiff(int [] arr,int n,int k)

{

int min=0,max=0,r=0;

Arrays.sort(arr);

r=arr[n-1]-arr[0];

for(int i=1;i<n;i++)

{

if(arr[i]>=k)

{

max=Math.max(arr[i-1]+k, arr[n-1]-k);

min=Math.min(arr[i]-k,arr[0]+k);

r=Math.min(r, max-min);

}

else

{

continue;

}

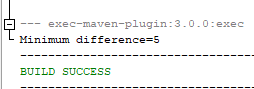
}

return r;

}

}

Output:



Question: Find the Duplicate array. Array in sorted order. And array include array[1,n+1] its means array not contain the negative elements. And its have only one repeated element and return the repeated element.

Solution:

class Find\_Duplicate{

public int Duplicate\_Element(int [] arr1,int length)

{

int index=0;

for(int i=0;i<arr1.length;i++)

{

index=Math.abs(arr1[i])-1;

if(arr1[index]<0)

return Math.abs(arr1[i]);

else

arr1[i]=-arr1[i];

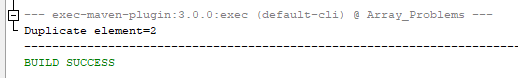
}

return -1;

}

}

Output:



Question: Find the jump to reach the end of the array.

Solution:

Its simple problem where we start from the zeroth index and search for the max no. which jump to closed to the array end.

class Find\_Jump

{

int max=0,jump=0,halt=0;

public int find\_jump(int [] arr,int length)

{

if(arr[0]==0)

{

return -1;

}

for(int i=0;i<arr.length;i++)

{

max=Math.max(max,i+arr[i]);

if(max>=arr.length-1)

{

jump+=1;

return jump;

}

if(i==halt)

{

halt=max;

jump++;

}

if(halt>=arr.length-1)

{

return jump;

}

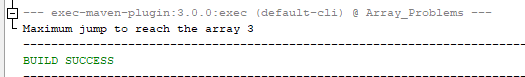
}

return -1;

}

}

Output:



Question: Merge the two sorted array.

Solution:

class Merge\_Sorted\_Array{

public void sorted\_array(int []arr1,int []arr2,int n,int m)

{

int i=0,j=0,k=0;

int [] merge\_sort=new int[n+m];

while(i<n && j<m)

{

if(arr1[i]>arr2[j])

{

merge\_sort[k]=arr2[j];

k++;

j++;

}

else if(arr1[i]<arr2[j])

{

merge\_sort[k]=arr1[i];

k++;

i++;

}

else

{

merge\_sort[k]=arr1[i];

k++;

i++;

j++;

}

}

while(i<n)

{

merge\_sort[k]=arr1[i];

k++;

i++;

}

while(j<m)

{

merge\_sort[k]=arr2[j];

k++;

j++;

}

System.out.println("After merging the two sorted array!!!");

for(int x=0;x<merge\_sort.length;x++)

{

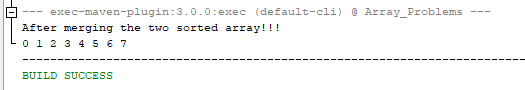
System.out.print(merge\_sort[x]+" ");

}

}

}

Output:



Question: Merge Interval.

Solution:

class Merge\_Interval{

public int [][] merge(int[][]interval)

{

//sorting

int flag=0;

for(int j=0;j<interval.length;j++)

{

if(interval[j][0]>interval[j+1][0])

{

//swap

int[] temp=new int[2];

temp[0]=interval[j][0];

temp[1]=interval[j][1];

interval[j][0]=interval[j+1][0];

interval[j][1]=interval[j+1][1];

interval[j+1][0]=temp[0];

interval[j+1][1]=temp[1];

flag=1;

}

if(flag==0)

break;

}

List <int[]>res=new ArrayList<>();

int current\_interval[]=interval[0];

res.add(current\_interval);

for(int i=0;i<interval.length;i++)

{

if(interval[i][0]<=current\_interval[1])

{

current\_interval[1]=Math.max(current\_interval[1], interval[i][1]);

}

else

{

current\_interval=interval[i];

res.add(current\_interval);

}

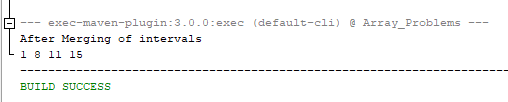
}

return res.toArray(new int [res.size()][]);

}

}

Output:



Question: Best time to buy or sell the stock.

Solution:

class Stock{

public int BUY\_SELL(int[]arr)

{

int profit=0;

int cur\_profit=0;

for(int i=0;i<arr.length;i++)

{

for(int j=i+1;j<arr.length;j++)

{

cur\_profit=arr[j]-arr[i];

if(cur\_profit>profit)

{

profit=cur\_profit;

}

}

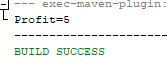
}

return profit;

}

}

Output:



Question: find the pair with the given sum.

Solution:

class pair{

public int count\_pair(int [] arr,int k)

{

int sum=0,count=0;

for(int i=0;i<arr.length;i++)

{

sum=0;

for(int j=i+1;j<arr.length;j++)

{

sum=arr[i]+arr[j];

if(sum==k)

count++;

}

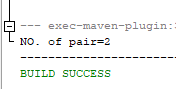
}

return count;

}

}

Output:



Question: Count Inversion.

Solution:

class count\_inversion{

public int count(int [] arr)

{

int inversion=0;

for(int i=0;i<arr.length;i++)

{

for(int j=i+1;j<arr.length;j++)

{

if(arr[i]>arr[j])

inversion++;

}

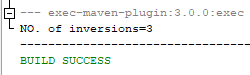
}

return inversion;

}

}

Output:



Question: find Intersection between two array:

Solution: Brute-Force approach.

Here Problem cleary define source Love babbar array tutorial.

class Find\_insertion{

public ArrayList<Integer> find\_insertion(int [] arr1,int [] arr2)

{

ArrayList<Integer> al= new ArrayList();

for(int i=0;i<arr1.length;i++)

{

for(int j=0;j<arr2.length;j++)

{

if(arr1[i]<arr2[j])

break;

if(arr1[i]==arr2[j])

{

al.add(arr1[i]);

arr1[j]=-1;

break;

}

}

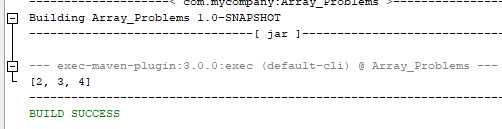
}

return al;

}

}

Output:



Question : Alternate Swaping of the array element.

Solution:

class Swap\_Alternate\_NO

{

public int [] Swap(int [] arr)

{

for(int i=0;i<arr.length;i+=2)

{

int temp=arr[i];

arr[i]=arr[i+1];

arr[i+1]=temp;

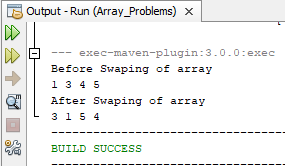
}

return arr;

}

}

Output:



Question: Find the unique no. of array.

Solution: We Used The XOR operator.

class find\_unique\_no

{

public int unique(int [] arr)

{

int ans=0;

for(int i=0;i<arr.length;i++)

{

ans=ans^arr[i];

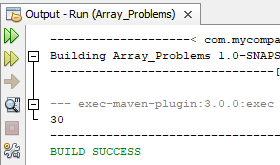
}

return ans;

}

}

Output



**Searching and Sorting Algorithm**

1. **Searching: Searching is the technique used for search the elements in the array.**
2. **There are different searching technique.**
3. **Linear Search**
4. **Binary Search**

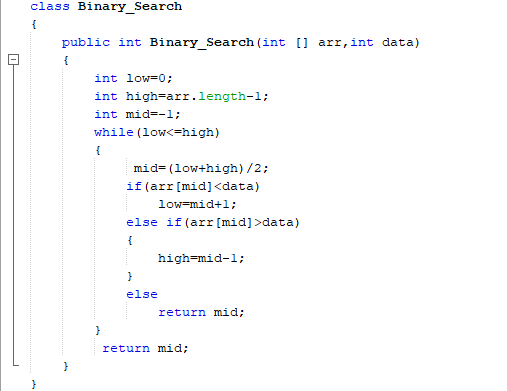
**Etc..**

**Linear Search:**

1. **Linear Search is search the element in array one by one.**
2. **The time complexity of linear search is O(n).**

**Binary Search**

1. **For binary search array should be in the sorted order.**
2. **In binary search Time complexity is O(log n).**

****

**In Binary search takes the three point low, mid, high:**

**Low point to the lower bound of the array**

**Mid point to the mid of the array**

**High point to the last index of the array**

**Sorting:**

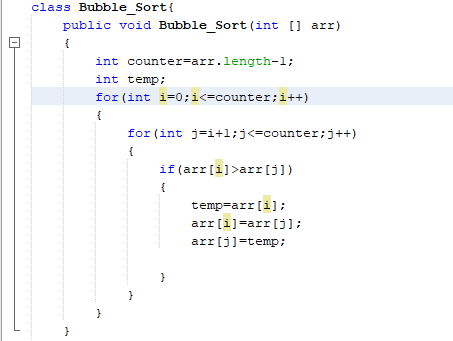
1. **Bubble Sort**
2. **Selection Sort**
3. **Insertion Sort**
4. **Merge Sort**
5. **Quick Sort**
6. **Heap Sort**

**All of this techniques used for sort the particular data structure.**

**But there is only difference in Time Complexity.**

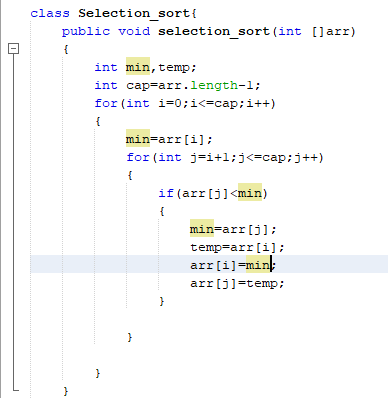
**Those takes less time that algorithm is the good algo.**

1. **Bubble Sort: In this algo try to put biggest no. first one there right place in one iteration one element reach there correct position.**

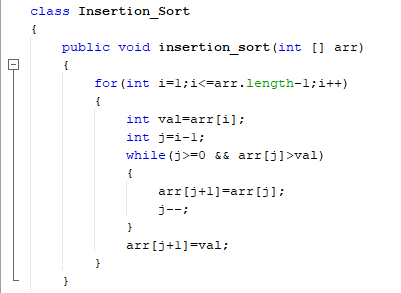
****

1. **Selection Sort:**
2. **In this algo select the min value and find out its right position.**

**Algorithm:**

****

1. **Insertion Sort:**
2. **In this sorting algo divide the array in two parts one is sorted and other is unsorted.**

****

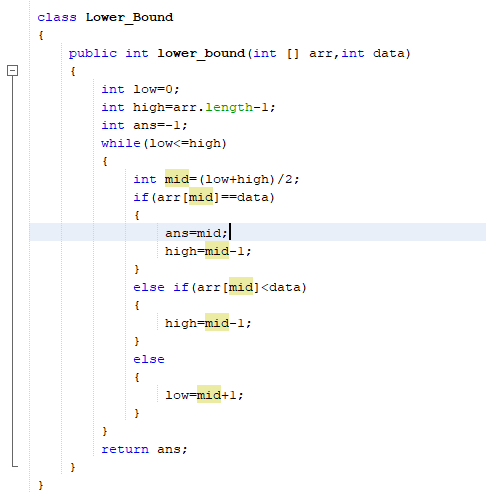
**Q1. Lower Bound of array element find out:**

**Arr[1,2,2,2,3,3,9,11]**

**Input:2**

**Output:1 (index no.)**

**Solve using the binary search:**

****

**Calculate till high not cross the low.**

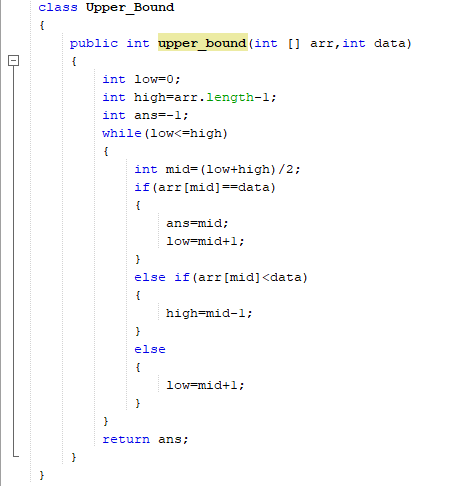
**Q2. Find the upper bound of array element**

**Arr[1,2,2,2,3,3,9,11]**

**Input:2**

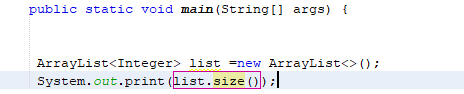
**Output:3 (index no.)**

**Solve using the binary search:**

****

**ArrayList**

1. **Array list is an array but the difference it array size is limited but the arraylist can grow its size at runtime.**
2. **Arraylist default capacity is 2.**
3. **Arraylist it is a class present in the java.util.ArrayList package.**
4. **If list is empty the size is 0**

****

****

1. **Add the element in array using add function.**
2. **If arraylist size==default capacity then arraylist grow.**

**Formula**

**size= current\_capacity\*2;**

10

20

**101 current array[2]**

**0 1**

**list.add(30)**

**Arraylist grow its size=2\*2=4**

**New arraylist created and reference stared to point new arraylist.**

30

10

10

**0 1 2 3**

1. **Access the element using the get function.**

**List.get(0) Index**

**10**

**List.get(3)**

**Error**

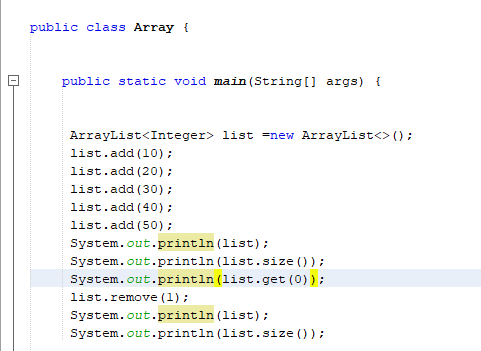
**Because at 3rd position no element it present and we can’t access the array out of its range in java.**

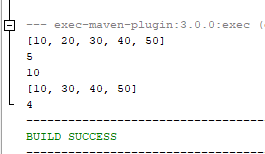
1. **We can remove the element of the list using remove function.**

**List.remove(2); index.**

**Removing element also decreases the arraylist size.**

1. **For printing the arraylist element not need the loops.**

****

****

**Q1. Spiral Matrix:**

**Print the 2D Matrix in the spiral form.**

**Solution:**

**class Sprial\_Matrix{**

**public void spiral\_matrix(int[][]arr)**

**{**

**int top=0;**

**int bottom=arr.length-1;**

**int left=0;**

**int right=arr[top].length-1;**

**int dir=1;**

**int count=((bottom+1)\*(right+1));**

**while(left<=right && top<=bottom)**

**{**

**if(count>0)**

**{**

**if(dir==1)**

**{**

**for(int i=left;i<=right;i++)**

**{**

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

**count--;**

**}**

**dir=2;**

**top++;**

**}**

**}**

**if(count>0)**

**{**

**if(dir==2)**

**{**

**for(int i=top;i<=bottom;i++)**

**{**

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

**count--;**

**}**

**dir=3;**

**right--;**

**}**

**}**

**if(count>0)**

**{**

**if(dir==3)**

**{**

**for(int i=right;i>=left;i--)**

**{**

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

**count--;**

**}**

**dir=4;**

**bottom--;**

**}**

**}**

**if(count>0)**

**{**

**if(dir==4)**

**{**

**for(int i=bottom;i>=top;i--)**

**{**

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

**count--;**

**}**

**dir=1;**

**left++;**

**}**

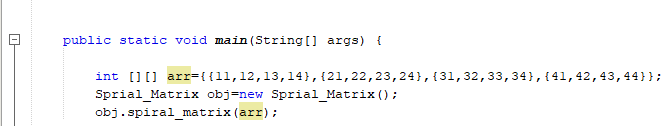
**}**

**}**

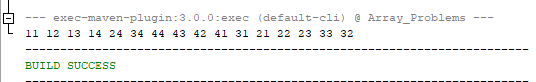
**}**

**}**

**Input:**

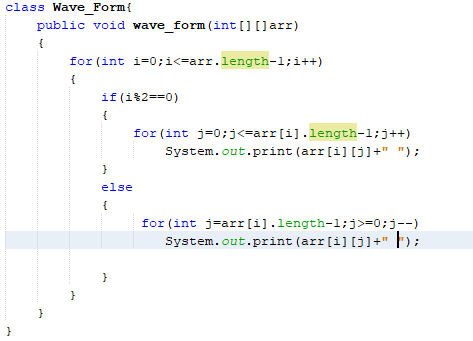
****

**Output:**

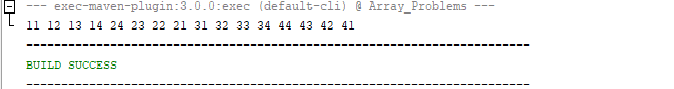
****

**Q. Print the 2D Matrix in wave form.**

**Solution:**

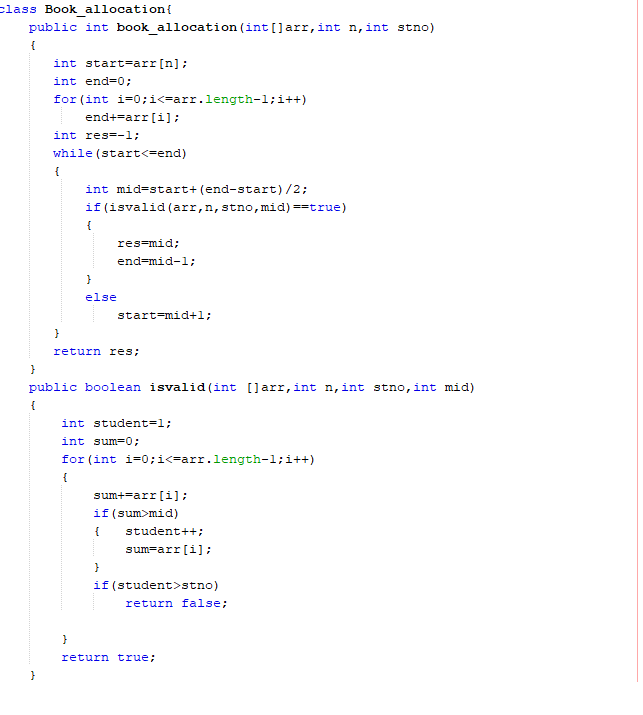
****

**Output:**

****

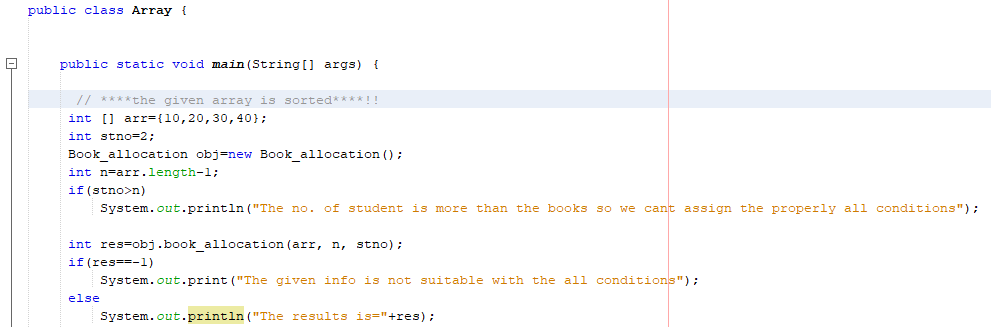
**Q. Book allocation Problem very famous problem(VVVIMP):**

**Solution:**

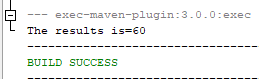
****

**This problem based on the binary search.**

**Input:**

****

Output:

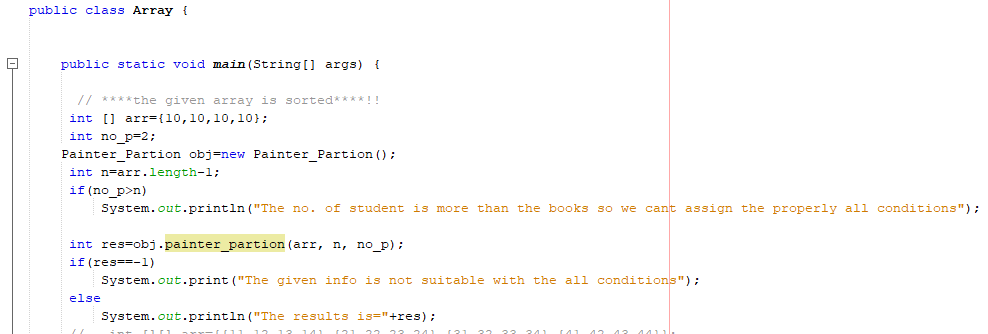


Q. Painter Partition Problem

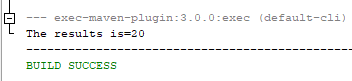
Solution:



Input:



Output:

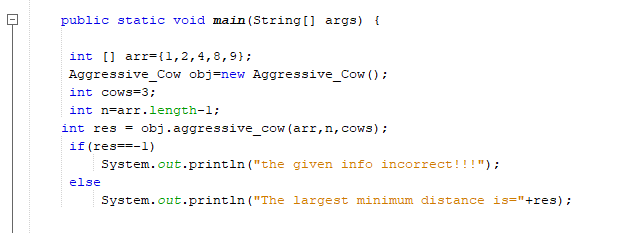


Q. Aggressive cow find the largest minimum distance

Solution:



Input:



Output:

