Lab3:

## Remove array element

public void remove(int a[],int index){ //by shifting left  
 if(index>=0 && index<a.length){  
 int i=index;  
 while (i<a.length-1){  
 a[i]=a[i+1];  
 i++;  
 }  
 a[i]=0;  
 }  
}

## Insertion Sort

public void insertionsort(int a[]){  
 for (int i=1;i<a.length;i++){ //تنتقل على جميع العناصر ومن 1 علشان نقارن مع الي قبل  
 int cur=a[i];  
 int j=i;  
 while(j>0&&cur<a[j-1]) //shifting loop  
 {  
 a[j]=a[j-1];  
 j--;  
 }  
 a[j]=cur;  
 }  
}

## Sequential search

public int seqsearch(int a[],int value){  
 int i=0;  
 int index=-1;  
 boolean flag=false;  
 while(i<a.length&&!flag){ //or flag==false  
 if(a[i]==value){  
 flag=true;  
 index=i;  
 }  
 i++;  
 }  
 return index;  
}

## java.util Methods for Arrays

* Arrays.equals(A,B)
* Arrays.fill(A,x)
* Arrays.copyOf(A,n)
* Arrays.coptOfRange(A,s,t)
* Arrays.sort(A)
* Arrays.binarySearch(A,x)
* import java.util.Arrays;
* /\*System.out.println(Arrays.equals(o.q,o.f));  
    
   o.f=Arrays.copyOf(o.q,o.q.length);  
    
   Arrays.fill(o.f,1);  
   System.out.println(Arrays.toString(o.f));  
    
   System.out.println(Arrays.toString(o.q));  
   Arrays.sort(o.q);  
   System.out.println(Arrays.binarySearch(o.q,114));\*/

## Fill array with Random Number

import java.util.Random;

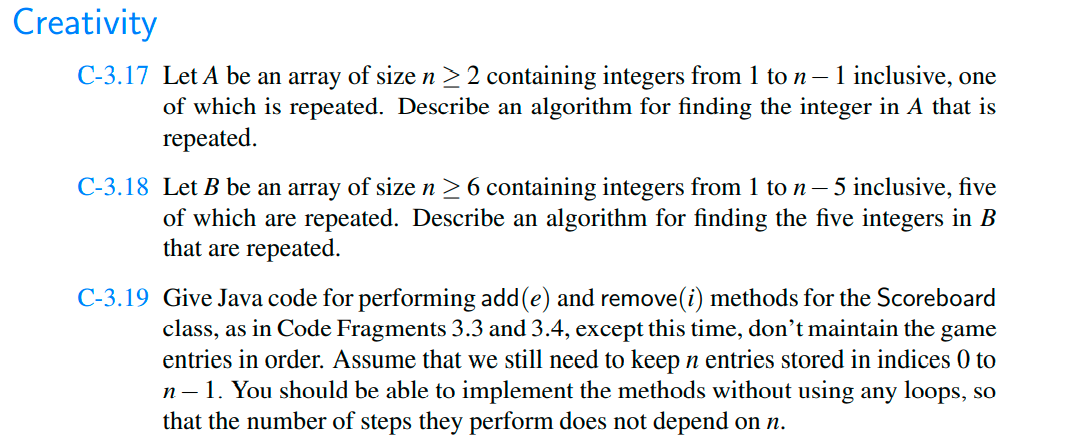
public void fillrandom(){  
 Random r=new Random();  
 r.setSeed(System.*currentTimeMillis*());  
 for(int i=0;i<q.length;i++){  
 q[i]=r.nextInt(10);  
 }  
}

## Convert String to char array

toCharArray

new String(char\_array)

Lab3 Tasks:



We know [sum of first n-1 natural numbers](https://www.geeksforgeeks.org/program-find-sum-first-n-natural-numbers/)is (n – 1)\*n/2. We compute sum of array elements and subtract natural number sum from it to find the only missing element.

3.17:

public class Tasks3 {  
  
  
 int num = 1;  
  
 public int repeate(int []N) {  
  
 int index=0;  
 boolean flag=false;  
  
  
 for(int j=0;j<N.length;j++) {  
 int i=j+1;  
 while (i < N.length && !flag) { //or flag==false  
 if (N[i] == N[j]) {  
 flag = true;  
 num++;  
 }  
 i++;  
 }  
  
 if (flag = true){index++;}  
 else {num=0;}  
 }  
 return index;  
  
 }  
  
 static int findRepeating(int[] arr, int n)  
 {  
 // Find array sum and subtract sum  
 // first n-1 natural numbers from it  
 // to find the result.  
  
 int sum = 0;  
 for (int i = 0; i < n; i++)  
 sum += arr[i];  
 return sum - (((n - 1) \* n)/2 );  
 }  
  
  
 public static void main(String[] args) {  
 Tasks3 o=new Tasks3();  
 int[] A = {1, 1};  
 int[] B = {5,5,5,5,5};  
  
  
 System.*out*.println("there are "+o.num+  
 " of integers are repeated "+o.repeate(A)+" times in the array:");  
  
 System.*out*.println(o.*findRepeating*(A,A.length));  
  
  
 /\* System.out.println("there are "+o.num+  
 " of integers are repeated "+o.repeate(B)+" times in the array:");  
  
 System.out.println(o.findRepeating(B,B.length));\*/  
  
 }  
}

Output:

there are 1 of integers are repeated 2 times in the array:

1

3.18: Only this method has changed:

static int findRepeating(int[] arr, int n)  
{  
 // Find array sum and subtract sum  
 // first n-1 natural numbers from it  
 // to find the result.  
  
 int sum = 0;  
 for (int i = 0; i < n; i++)  
 sum += arr[i];  
 return sum - (((n - 1) \* n) );  
}

System.*out*.println("there are "+o.num+  
 " of integers are repeated "+o.repeate(B)+" times in the array:");  
  
 System.*out*.println(o.*findRepeating*(B,B.length));  
  
}

Output:

there are 1 of integers are repeated 5 times in the array:

5

The Game : (NOT COMPLETED)

public class Game {  
   
 public class GameEntry{  
 private String name ;  
 private int score;  
  
 public GameEntry(String name, int score) {  
 this.name = name;  
 this.score = score;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public int getScore() {  
 return score;  
 }  
  
 @Override  
 public String toString() {  
 return "GameEntry{" +  
 "name='" + name + '\'' +  
 ", score=" + score +  
 '}';  
 }  
 }  
   
 public class Scoreboard{  
   
 private int numEntries =0;  
 private GameEntry[] board;  
   
 public Scoreboard(int capacity){  
 board=new GameEntry[capacity];  
 }  
   
 public void add(GameEntry e){  
 int newScore =e.getScore();  
   
 if (numEntries<board.length||newScore>board[numEntries-1].getScore()){  
 if(numEntries<board.length)  
 numEntries++;  
   
 int j=numEntries-1;  
 while(j>0&&board[j-1].getScore()<newScore){  
 board[j]=board[j-1];  
 j--;  
 }  
 board[j]=e;  
 }  
   
 }  
   
   
 public GameEntry remove(int i)throws IndexOutOfBoundsException{  
 if(i<0||i>numEntries)  
 throw new IndexOutOfBoundsException("invaled:"+i);  
   
 GameEntry temp =board[i];  
   
 for(int j=i;j<numEntries-1;j++){  
 board[j]=board[j+1];  
 }  
   
 board[numEntries-1]=null;  
 numEntries--;  
 return temp;  
 }  
   
 }  
   
   
}