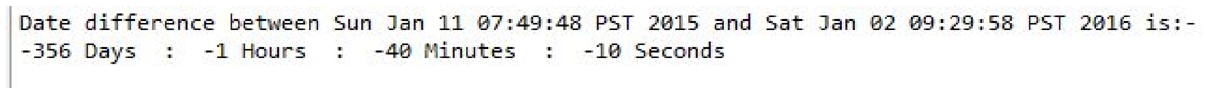
1. Find out the number of days in between two given dates ?

```
public class DateDifference {

  public static void main(String[] args) throws ParseException {
        // TODO Auto-generated method stub
        SimpleDateFormat format = new SimpleDateFormat("mm-dd-yy hh:mm:ss");
        //Date date1 =new Date("12-02-2014");
        //Date date2 =new Date("22-02-2014");
        Date dateA = format.parse("10-11-2015 07:49:48");
        Date dateB = format.parse("12-02-2016  09:29:58");
        long difference = dateA.getTime()- dateB.getTime();
        long diffSeconds = difference / 1000;
        long diffMinutes = difference / (60 * 1000);
        long diffHours = difference / (60 * 60 * 1000);
        long diffDays = difference / (24* 60 * 60 * 1000);
        System.out.println("Date difference between "+dateA +" and "+dateB +" is:-");
        System.out.print(diffDays+" Days ");
        System.out.print(" :  "+diffHours%24+" Hours ");
        System.out.print(" :  "+diffMinutes%60+" Minutes ");
        System.out.print(" :  "+diffSeconds%60+" Seconds ");


  }

}
```

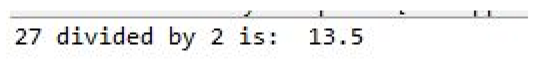


```
Date difference between Sun Jan 11 07:49:48 PST 2015 and Sat Jan 02 09:29:58 PST 2016 is:-
-356 Days  :  -1 Hours  :  -40 Minutes  :  -10 Seconds
```

2. How to divide a number by 2 without using / operator?

```
public class DivideBy2woOperand {

  public static void main(String[] args) {
        // TODO Auto-generated method stub
int num = 27;
double half= 0;
int counter = num;


for (int i = 0; i<=num;i++){
  if (num==1){
        half = 0.5;
```

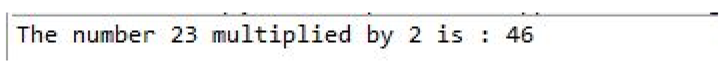
```
            break;
    }
    if (num==2){
            half = 1.0;
            break;
    }
    if (counter==i){
            half = i;
            break;
    }
    if (counter==i+1){
            half = i+0.5;
            break;
    }
    counter--;
}

System.out.println(num +" divided by 2 is:  "+half);
    }

}
```



```
27 divided by 2 is:  13.5
```

3. How to multiply a number by 2 without using * operator?

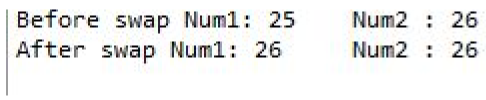
```
public class MultiplyWith2woOperand {

    public static void main(String[] args) {
            // TODO Auto-generated method stub
int num = 23;
int myDouble = num+num;
System.out.println("The number "+num+" multiplied by 2 is : "+myDouble);
}
}
```



```
The number 23 multiplied by 2 is : 46
```

4. How to swap two variables, by using pass by reference method ?

```
public class WrapInt {
    private int value;

    public WrapInt(int value) {
            super();
            this.value = value;
    }

    public WrapInt(WrapInt wrapInt) {
            // TODO Auto-generated constructor stub
            this.value= wrapInt.getValue();
    }

    public int getValue() {
            return value;
    }

    public void setValue(int value) {
            this.value = value;
    }

    @Override
    public String toString() {
            // TODO Auto-generated method stub
            return String.valueOf(value);
    }

    public static void main(String[] args) {
            // TODO Auto-generated method stub
            WrapInt num1 = new WrapInt(25);
            WrapInt num2 = new WrapInt(26);
            System.out.println("Before swap Num1: "+num1+ "\tNum2 : "+num2);
            swapVariables(num1 ,num2);
            System.out.println("After swap Num1: "+num1+ "\tNum2 : "+num2);
    }



    private static void swapVariables(WrapInt wrapInt1, WrapInt wrapInt2) {
            // TODO Auto-generated method stub
            wrapInt1.setValue( wrapInt2.getValue());
            wrapInt2.setValue(wrapInt2.getValue());
    }
}
```



```
Before swap Num1: 25    Num2 : 26
After swap Num1: 26     Num2 : 26
```

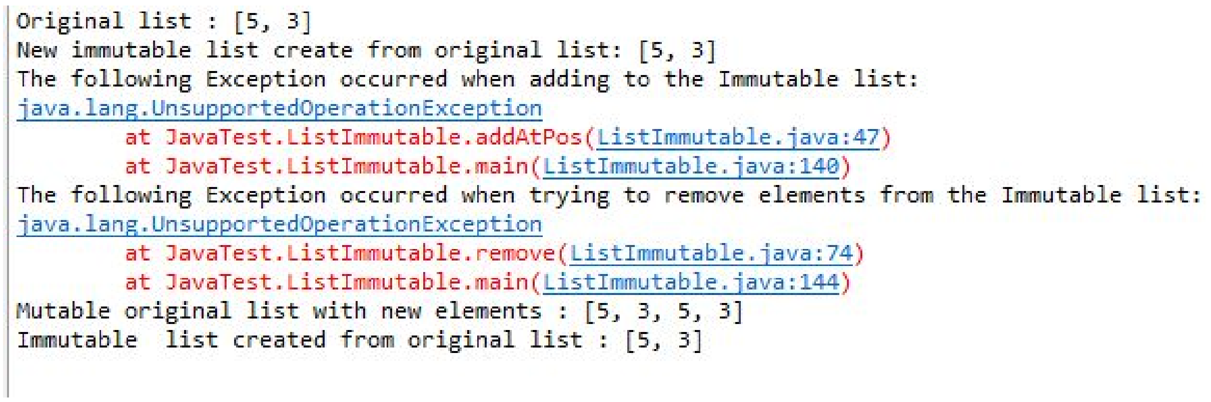
5. How to make a list immutable?

```
public class ListImmutable<E> extends LList<E>{
 private  final Node start;
 private  final Node end;
 private  final int  size;
 final LList<E> list;

    public ListImmutable(LList<E> list1) {
            super();
            // TODO Auto-generated constructor stub
            this.list=list1;
            this.start= list.getStart();
            this.end=list1.getEnd();
            this.size=list1.size;

    }

    @Override
    public Node getStart() {
            return start;
    }
    @Override
    public Node getEnd() {
            return end;
    }

    @Override
    public void setEnd(Node end) {
            // TODO Auto-generated method stub
            try {
                    throw new UnsupportedOperationException();
            } catch (UnsupportedOperationException e) {
                    // TODO Auto-generated catch block
                    String exceptionMsg = "The following Exception occurred when modifying  the Immutable
list: ";
                    System.out.println(exceptionMsg);
                    e.printStackTrace();

            }
    }
    @Override
    public void addAtPos(int index, int data) {
            // TODO Auto-generated method stub
            try {
                    throw new UnsupportedOperationException();
```

```
		} catch (UnsupportedOperationException e) {
			// TODO Auto-generated catch block
			String exceptionMsg = "The following Exception occurred when adding to the Immutable
list: ";
			System.out.println(exceptionMsg);
			e.printStackTrace();

		}

	}
	@Override
	public void add(int data) {
		// TODO Auto-generated method stub
		try {
			throw new UnsupportedOperationException();
		} catch (UnsupportedOperationException e) {
			// TODO Auto-generated catch block
			String exceptionMsg = "The following Exception occurred when modifying to the Immutable
list: ";
			System.out.println(exceptionMsg);
			e.printStackTrace();

		}
	}
	@Override
	public void remove(int data) {
		// TODO Auto-generated method stub
		try {
			throw new UnsupportedOperationException();
		} catch (UnsupportedOperationException e) {
			// TODO Auto-generated catch block
			String exceptionMsg = "The following Exception occurred when trying to remove elements
from the Immutable list: ";
			System.out.println(exceptionMsg);
			e.printStackTrace();

		}
	}
	@Override
	public int getSize() {
		// TODO Auto-generated method stub
		return size;
	}
	@Override
	public void setSize(int size) {
		// TODO Auto-generated method stub
		try {
			throw new UnsupportedOperationException();
```

```
            } catch (UnsupportedOperationException e) {
                        // TODO Auto-generated catch block
                        String exceptionMsg = "The following Exception occurred when modifying  the Immutable
list: ";
                        System.out.println(exceptionMsg);
                        e.printStackTrace();

            }
    }
    @Override
    public void setStart(Node start) {
            // TODO Auto-generated method stub
            try {
                        throw new UnsupportedOperationException();
            } catch (UnsupportedOperationException e) {
                        // TODO Auto-generated catch block
                        String exceptionMsg = "The following Exception occurred when modifying to the Immutable
list: ";
                        System.out.println(exceptionMsg);
                        e.printStackTrace();

            }

    }
    @Override
    public int[] toArray() {
            // TODO Auto-generated method stub
            Node ptr = start;
            int[] array = new int[size];
            for(int i =0;i<size;i++){
                        array[i]=ptr.getData();
                        ptr=ptr.getLink();
            }
            return array;

    }

    public static void main(String[] args) {
            // TODO Auto-generated method stub
            // TODO Auto-generated method stub
                                    LList<Integer> list = new LList<Integer>();
                                    list.addAtPos(1, 3);
                                    list.addAtPos(1,5);
                                    //list = Collections.unmodifiableList(list);
                                    //list.remove(1); //get Exception
                                    System.out.println("Original list : "+Arrays.toString(list.toArray()));
                                    ListImmutable<Integer> finallist=  new ListImmutable<Integer>(list);
```

```
                              System.out.println("New immutable list create from original list:
"+Arrays.toString(finallist.toArray()));
                              try{
                                        finallist.addAtPos(3,8);// get exception
                              }catch(Exception e){
                                        e.printStackTrace();
                              }
                              finallist.remove(1);
                              list.addAtPos(1, 3);
                              list.addAtPos(1,5);
                              System.out.println("Mutable original list with new elements :
"+Arrays.toString(list.toArray()));

                              System.out.println("Immutable  list created from original list :
"+Arrays.toString(finallist.toArray()));
                              //finallist.remove(1);

   }
   }
```



```
Original list : [5, 3]
New immutable list create from original list: [5, 3]
The following Exception occurred when adding to the Immutable list:
java.lang.UnsupportedOperationException
        at JavaTest.ListImmutable.addAtPos(ListImmutable.java:47)
        at JavaTest.ListImmutable.main(ListImmutable.java:140)
The following Exception occurred when trying to remove elements from the Immutable list:
java.lang.UnsupportedOperationException
        at JavaTest.ListImmutable.remove(ListImmutable.java:74)
        at JavaTest.ListImmutable.main(ListImmutable.java:144)
Mutable original list with new elements : [5, 3, 5, 3]
Immutable  list created from original list : [5, 3]
```

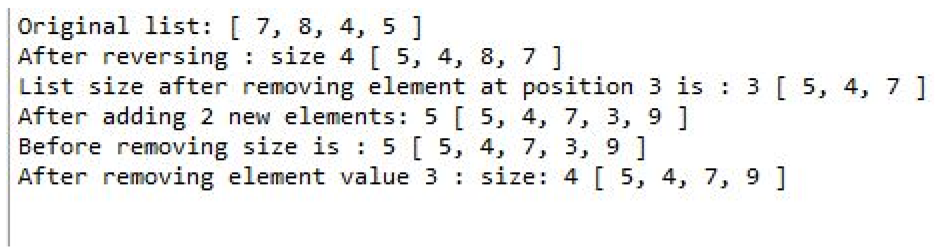
6. Write a sample code to reverse Singly Linked List by iterating through it only once.

```
package JavaTest;

public class LList<E> {
   private Node start;
   private Node end;
   int size;
   public Node getStart() {
         return start;
   }
```

```
public Node getEnd() {
        return end;
}
public void setEnd(Node end) {
        this.end = end;
}
public void addAtPos(int index, int data){
        size++;
        if(index == 1){
                if(start == null){
                        Node newNode = new Node(data, null);
                        start = newNode;
                        end = start;
                        end.setLink(null);
                }else{
                        Node newHead = new Node(data, start);
                        start=newHead;
                }
        }
        index = index-1;
        Node newNode = new Node(data, null);
        Node ptr = start;
        for(int i = 1;i<size;i++){
                if (i==index){
                        Node tmp = ptr.getLink();
                        ptr.setLink(newNode);
                        newNode.setLink(tmp);
                }
                ptr=ptr.getLink();
        }

}
public void add(int data){
        Node ptr = new Node(data, null);
        end.setLink(ptr);
        //System.out.println(end.getData()+ end.getLink().getData());
        end = ptr;
        size++;
}

private void show() {
        // TODO Auto-generated method stub
        Node ptr= start;
        System.out.print("[ ");
        while(ptr.getLink()!=null){
                System.out.print(ptr.getData()+", ");
                ptr=ptr.getLink();
        }
        System.out.println(ptr.getData()+" ]");
}
private void show(Node head) {
        // TODO Auto-generated method stub
        Node ptr= head;
        System.out.print("[ ");
        while(ptr.getLink()!=null){
                System.out.print(ptr.getData()+", ");
                ptr=ptr.getLink();
        }
        System.out.println(ptr.getData()+" ]");

}

private Node reverse(Node head){
        Node previous = null;
        Node current = head;
        end =head;
        Node next = null;
        Node tmp =null;
        while(current.getLink().getLink()!=null){

                next = current.getLink();
                current.setLink(previous);
                tmp =  next.getLink();

                next.setLink(current);

                previous = next;
                current = tmp;
        }
        next = current.getLink();
        next.setLink(current);
        current.setLink(previous);
        start=next;
        //end=head;
        return next;
}

private void removeAtPos(int position){

        if(position == 0){
                start= start.getLink();
                return;

        }
```

```
		position =position-1;
		Node ptr = start;
		//Node next = ptr.getLink();
		for(int i = 1; i<size;i++){
			if(i==position){

				Node tmp = ptr.getLink().getLink();
				ptr.setLink(tmp);
				if(i==size-1){
					end =ptr;
				}
				break;
			}

			ptr=ptr.getLink();
		}
		size--;
	}

	public void remove(int data){
		size--;
		Node ptr = start;

		for(int i = 1; i<size;i++){
			if(ptr.getLink().getData()==data){
				Node tmp = ptr.getLink().getLink();
				ptr.setLink(tmp);
				break;
			}
			ptr=ptr.getLink();
		}
	}

	public static void main(String[] args) {
		LList<Integer> list = new LList<Integer>();
		list.addAtPos(1,5);
		list.addAtPos(1,7);
		list.addAtPos(2, 4);
		list.addAtPos(2, 8);

		System.out.print("Original list: ");list.show();
		list.reverse(list.getStart());
		System.out.print("After reversing : size "+list.size+" ");list.show();

		list.removeAtPos(3);
		System.out.print("List size after removing element at position 3 is : "+list.size+" ");list.show();
		list.add(3);
		list.add(9);
```

```
        System.out.print("After adding 2 new elements: "+list.size+" ");list.show();
        System.out.print("Before removing size is : "+list.size+" ");list.show();
        list.remove(3);

        System.out.print("After removing element value 3 : size: "+list.size+" ");list.show();
    }

    public int getSize() {
        return size;
    }

    public void setSize(int size) {
        this.size = size;
    }

    public void setStart(Node start) {
        this.start = start;
    }

    public int[] toArray() {
        // TODO Auto-generated method stub
        Node ptr = start;
        int[] array = new int[size];
        for(int i =0;i<size;i++){
            array[i]=ptr.getData();
            ptr=ptr.getLink();
        }
        return array;

    }
}
```



```
Original list: [ 7, 8, 4, 5 ]
After reversing : size 4 [ 5, 4, 8, 7 ]
List size after removing element at position 3 is : 3 [ 5, 4, 7 ]
After adding 2 new elements: 5 [ 5, 4, 7, 3, 9 ]
Before removing size is : 5 [ 5, 4, 7, 3, 9 ]
After removing element value 3 : size: 4 [ 5, 4, 7, 9 ]
```

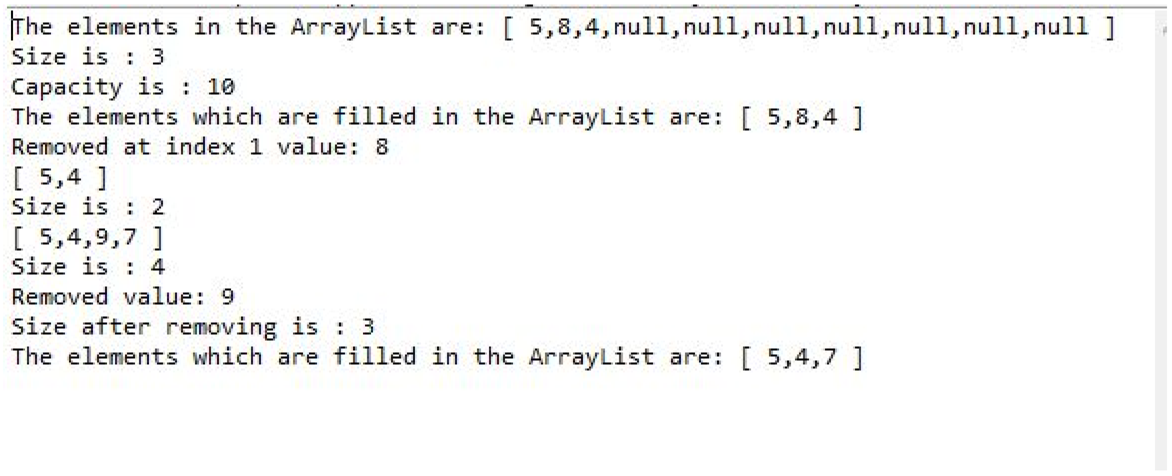
7. Write a program to implement ArrayList and Linked list

Linked List implementation as above

```
public class ArraylistImpl<E>{
    private Object[] elementData;
    private int size;


    public ArraylistImpl(int initialCapacity){
            this.elementData=new Object[initialCapacity];
    }
    public ArraylistImpl(){
            this(10);
    }
    public ArraylistImpl(Collection<? extends E>c){
            elementData = c.toArray();
            size = elementData.length;
            if(elementData.getClass()!=Object[].class)
                    elementData = Arrays.copyOf(elementData,size, Object[].class);
    }


    public void ensureCapacity(int minCapacity){
            int oldCapacity = elementData.length;

            if(minCapacity> oldCapacity){
                    int newCapacity = (oldCapacity*3)/2+1;
                    if(newCapacity<minCapacity)
                            newCapacity=minCapacity;
                    elementData=Arrays.copyOf(elementData,newCapacity);
            }
    }
    public boolean add(E e){
            ensureCapacity(size+1);
            elementData[size++]=e;
            return true;

    }

    public void add(int index, E element){
            ensureCapacity(size+1);
            System.arraycopy(elementData, index, elementData, index+1, size-index);
            elementData[index] =element;
            size++;
    }

    public boolean addAll(Collection<? extends E>c){
            Object[] a = c.toArray();
            int newNum = a.length;
            ensureCapacity(size+newNum);
            System.arraycopy(a, 0, elementData,size,newNum);
```

```
        size+=newNum;
        return newNum!=0;

}
public E remove(int value){
        E oldValue = null;
        for(int i = 0;i<size;i++){
                if(elementData[i] == Integer.valueOf(value)){

                         oldValue = elementData(i);
                        int numMoved =size-i-1;
                        if(numMoved > 0)
                                System.arraycopy(elementData, i+1, elementData, i, numMoved);
                        size--;
                        break;
                }
        }
        return oldValue;

}

public E removeAtIndex(int index){
        E oldValue = elementData(index);
        int numMoved =size-index-1;
        if(numMoved > 0)
                System.arraycopy(elementData, index+1, elementData, index, numMoved);
        size--;
        return oldValue;

}
private E elementData(int index) {
        // TODO Auto-generated method stub
        E e = (E) elementData[index];

        return e;
}

public void show(){
        System.out.print("[ ");
        for(int i =0;i<capacity()-1;i++){
                System.out.print(elementData[i]+",");
        }
        System.out.println(elementData[capacity()-1]+" ]");


}
public void show(ArraylistImpl<E> list){
        System.out.print("[ ");
```

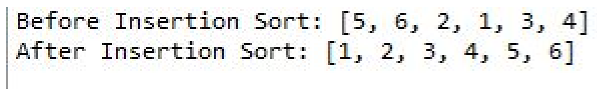
```
            for(int i =0;i<size-1;i++){
                    System.out.print(elementData[i]+",");
            }
            System.out.println(elementData[size-1]+" ]");

    }
    public int capacity() {
            // TODO Auto-generated method stub
            return elementData.length;
    }
    public int size() {
            // TODO Auto-generated method stub
            return size;
    }
    public static void main(String[] args) {
            // TODO Auto-generated method stub
            ArraylistImpl<Integer> alist = new ArraylistImpl<Integer>();
            alist.add(5);
            alist.add(8);
            alist.add(2, 4);
            System.out.print("The elements in the ArrayList are: ");alist.show();
            //Shows elements in index capacity
            System.out.println("Size is : "+alist.size());
                                        //insertion order preserved
            System.out.println("Capacity is : "+alist.capacity());
            System.out.print("The elements which are filled in the ArrayList are: ");alist.show(alist);    //Shows
elements in size
            int a = alist.removeAtIndex(1);
            System.out.println("Removed at index 1 value: "+a);
            alist.show(alist);
            System.out.println("Size is : "+alist.size());
                                        //removes at index and returns value
            alist.add(9);
            alist.add(7);
            alist.show(alist);
            System.out.println("Size is : "+alist.size());
            a = alist.remove(9);
                                                                            //removes a value

            System.out.println("Removed value: "+a);
            System.out.println("Size after removing is : "+alist.size());
            System.out.print("The elements which are in the ArrayList are: ");alist.show(alist);
    }


}
```


```
The elements in the ArrayList are: [ 5,8,4,null,null,null,null,null,null,null ]
Size is : 3
Capacity is : 10
The elements which are filled in the ArrayList are: [ 5,8,4 ]
Removed at index 1 value: 8
[ 5,4 ]
Size is : 2
[ 5,4,9,7 ]
Size is : 4
Removed value: 9
Size after removing is : 3
The elements which are filled in the ArrayList are: [ 5,4,7 ]
```

8. Write a program for Insertion Sort in java.

```
public class InsertionSortDemo {

  public static void main(String[] args) {
        // TODO Auto-generated method stub

        /*
         * Time complexity of selection sort is O(n2)
         */
        int[] a = {5,6,2,1,3,4};
        System.out.println("Before Insertion Sort: "+Arrays.toString(a));
        InsertionSort(a);
        System.out.println("After Insertion Sort: "+Arrays.toString(a));
  }

  static void InsertionSort(int[] a){


        int sizeOfList = a.length;
        for (int i = 0; i<sizeOfList-1 ;i++){
                for(int j = i+1; j<sizeOfList;j++){
                        if ( a[i] > a[j] ){
                                int temp = a[i];
                                a[i]= a[j];
                                a[j]= temp;

                        }

                }

        }
```

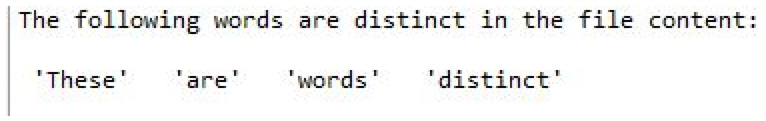
```
    }

}
```



```
Before Insertion Sort: [5, 6, 2, 1, 3, 4]
After Insertion Sort: [1, 2, 3, 4, 5, 6]
```

9. Write a program to get distinct word list from the given file.

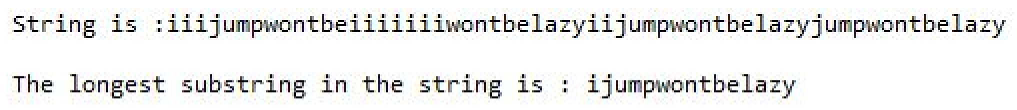
```
public class DistinctWordsInFile {

    public static void main(String[] args) throws FileNotFoundException {
        // TODO Auto-generated method stub
        FileInputStream fs = new FileInputStream("C:\\Users\\MadhuBindu\\Desktop\\File.txt");
        Scanner in = new Scanner(fs);
        String text = in.useDelimiter("\\A").next();
        in.close(); // Put this call in a finally block
        String[] strArray = text.replaceAll("\\W", " ").split(" ");

        HashMap<String, Integer> hashmap= new HashMap<String, Integer>();
        for (int i = 0; i<strArray.length-1;i++){
            int frequency=0;
            for (int j = 1; j<strArray.length;j++){
                if(strArray[i].equals(strArray[j])){
                    hashmap.put(strArray[i], ++frequency);

                }else{
                    if(j==strArray.length-1){
                        hashmap.put(strArray[j], 1);

                    }
                }

            }
        }
        Iterator<Entry<String, Integer>> it = hashmap.entrySet().iterator();
        System.out.println("The following words are distinct in the file content: ");
        System.out.println();
        while (it.hasNext()) {
            Map.Entry<String, Integer> keyValuepair = (Map.Entry<String, Integer>)it.next();
            if(keyValuepair.getValue()>=1){
                System.out.print( " '"+keyValuepair.getKey()+"'  ");
            }
        }
    }

}
```



```
The following words are distinct in the file content:

 'These'   'are'   'words'   'distinct'
```

10. Find longest substring without repeating characters.

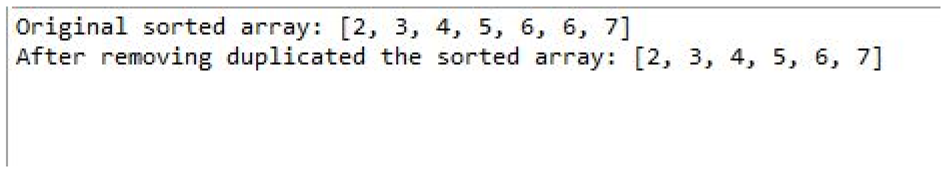
```
public class LongestSubstring {

    public static void main(String[] args){
            String str = "iiijumpwontbeiiiiiiiwontbelazyiijumpwontbelazyjumpwontbelazy";
            System.out.println("String is :"+str);
            String longest =getLongestSubString(str);
            System.out.println();
            System.out.println("The longest substring in the string is : "+longest);

    }
    private static String getLongestSubString(String str) {
            String charSubStr = "";
            String LongestSubStr = "";
            for (int i = 0; i < str.length(); i++) {
                    char c = str.charAt(i);
                    if (charSubStr.indexOf(c) == -1) {
                            charSubStr += c;
                            continue;
                    } else {
                            if(charSubStr.length()> LongestSubStr.length())
                                    LongestSubStr = charSubStr;
                            charSubStr = charSubStr.substring(charSubStr.indexOf(c) + 1) +c;
                    }

            }

            if(charSubStr.length() > LongestSubStr.length())

                    LongestSubStr = charSubStr;

            return LongestSubStr;

    }
}
```



```
String is :iiijumpwontbeiiiiiiiwontbelazyiijumpwontbelazyjumpwontbelazy

The longest substring in the string is : ijumpwontbelazy
```

11. Write a program to remove duplicates from sorted array

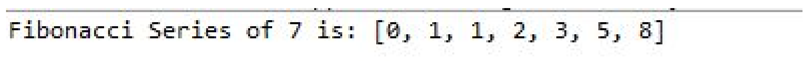
```
public class RemoveDupsInSortedArray {

    public static void main(String[] args) {
            // TODO Auto-generated method stub
            int[] myArray = {2,3,4,5,6,6,7};
            System.out.println("Original sorted array: "+Arrays.toString(myArray));
            myArray=removeDups(myArray);
            System.out.println("After removing duplicated the sorted array: "+Arrays.toString(myArray));
    }

    private static int[] removeDups(int[] myArray) {
            // TODO Auto-generated method stub
             int j = 0;
            int i = 1;
            if(myArray.length < 2){
            return myArray;
            }
            while(i < myArray.length){
            if(myArray[i] == myArray[j]){
                    i++;
            }else{
                    myArray[++j] = myArray[i++];
            }
            }
            int[] output = new int[j+1];
            for(int k=0; k<output.length; k++){
            output[k] = myArray[k];
            }

            return output;
    }
}
```



```
Original sorted array: [2, 3, 4, 5, 6, 6, 7]
After removing duplicated the sorted array: [2, 3, 4, 5, 6, 7]
```

12. Write a program to print fibonacci series.

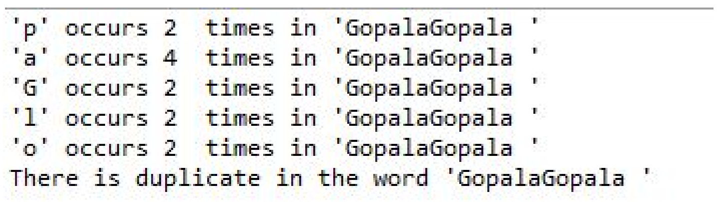
```
package JavaTest;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class TestFib {


        public static void main(String[] args) {
        // TODO Auto-generated method stub
        int num =7;

        List<Integer> fibonacciSeries = new ArrayList<Integer>();
        for(int i = 0;i<num;i++){
        int fibonacciValue = fibonacci(i);

                fibonacciSeries.add(i, fibonacciValue);
        }
        System.out.println("Fibonacci Series of "+num +" is: "+Arrays.toString(fibonacciSeries.toArray()));

        }
        private static int fibonacci(int i) {
        // TODO Auto-generated method stub
                if(i<=1)
                        return i;
                        else
                        return fibonacci(i-1) + fibonacci(i-2);
        }


  }
```



```
Fibonacci Series of 7 is: [0, 1, 1, 2, 3, 5, 8]
```

13. Write a program to find out duplicate characters in a string

```
public class FindDuplicate {

    public static void main(String[] args) {
            // TODO Auto-generated method stub
            HashMap<String, Integer> hashmap= new HashMap<String, Integer>();
            String str = "GopalaGopala ";
            for (int i = 0; i<str.length();i++){
                    int frequency=0;
                    for (int j = 0; j<str.length();j++){

                            if(str.charAt(i) == str.charAt(str.length()-j-1)){

                                    hashmap.put(String.valueOf(str.charAt(i)), ++frequency);
                            }
                    }
            }
            Iterator<Entry<String, Integer>> it = hashmap.entrySet().iterator();
            boolean hasDuplicate = false;
             while (it.hasNext()) {
                    Map.Entry<String, Integer> keyValuepair = (Map.Entry<String, Integer>)it.next();
                    if((int)keyValuepair.getValue()>1){
                            hasDuplicate= true;
                            System.out.println("'"+keyValuepair.getKey()+"' occurs "+keyValuepair.getValue()+
"  times in '"+str+"'"
                                            );
                    }
                                            }
             String result = hasDuplicate==false ? "There is no duplicate in the word '"+ str+ "'": "There is
duplicate in the word '"+ str+ "'";
             System.out.println(result);
    }

}
```

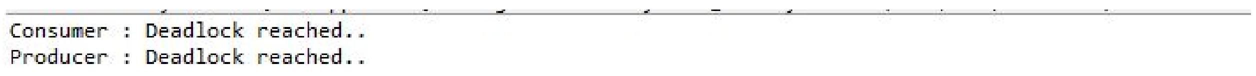


```
'p' occurs 2  times in 'GopalaGopala '
'a' occurs 4  times in 'GopalaGopala '
'G' occurs 2  times in 'GopalaGopala '
'l' occurs 2  times in 'GopalaGopala '
'o' occurs 2  times in 'GopalaGopala '
There is duplicate in the word 'GopalaGopala '
```

14. Write a program to create deadlock between two threads

```
    public class DeadLockDemo {

            public static void main(String a[]){
```

```
            // Test thread = new Test();
            ArrayList<Product> myproducts = new ArrayList<Product>();

                        Product product1 = new Product("iPhone", 2);
                        Product product2 = new Product("Samsung", 2);
                        myproducts.add(product1);
                        myproducts.add(product2);
                        Producer producer = new Producer(myproducts);

                        Consumer consumer = new Consumer(myproducts);

                        producer.start();
                        consumer.start();
            }
    }
```

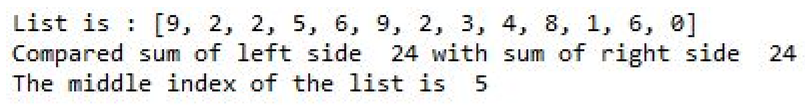


```
Consumer : Deadlock reached..
Producer : Deadlock reached..
```

15. Find out middle index where sum of both ends are equal

```
public class FindMedianIndexDemo {

    public static void main(String[] args) {
            // TODO Auto-generated method stub
            int[] list = {9,2, 2,5,6,9, 2,3,4, 8,1 ,6,0};
            System.out.println("List is : "+Arrays.toString(list));
            int  leftSum =list[0];

            for(int i = 1;i<list.length;i++){

                        int lastIndex = list.length-1;
                        int rightSum=list[lastIndex];

                        for(int j = lastIndex-1;j>i;j--){

                                    rightSum = rightSum+list[j];
                        }
                        if(rightSum ==leftSum ){



                                    System.out.println("Compared sum of left side  "+leftSum+" with sum of right side
"+rightSum);
```

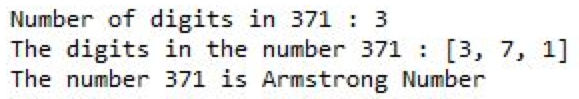
```
                    System.out.println("The middle index of the list is  "+i);
                    break;

                }

                if(i==list.length-2 ){
                    System.out.println("Did not find median index in the list");
                    break;
                }

                leftSum = list[i]+leftSum;
            }
    }

}
```



```
List is : [9, 2, 2, 5, 6, 9, 2, 3, 4, 8, 1, 6, 0]
Compared sum of left side  24 with sum of right side  24
The middle index of the list is  5
```

16. Write a program to find the given number is Armstrong number or not?

```
public class ArmstrongNumberDemo {

    public static void main(String[] args) {
            // TODO Auto-generated method stub
int num = 371;

String result = isArmstrong(num)?"The number "+num+" is Armstrong Number":"The number "+num+" is not
Armstrong Number";
System.out.println(result);
    }

    private static boolean isArmstrong(int num) {
            // TODO Auto-generated method stub
            int numOfDigits = String.valueOf(num).split("").length;
            String[] nums = String.valueOf(num).split("");
            int sumOfDigitsPower =0;
            System.out.println("Number of digits in "+num+" : "+numOfDigits);
            System.out.println("The digits in the number "+num+" : "+Arrays.toString(nums));
            for(int i = 0; i<numOfDigits;i++){
                    int digit = Integer.valueOf(nums[i]);
                    sumOfDigitsPower += Math.pow(digit, numOfDigits);
            }
            return sumOfDigitsPower ==num;
    }
}
```



```
Number of digits in 371 : 3
The digits in the number 371 : [3, 7, 1]
The number 371 is Armstrong Number
```