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

```
package javaTest;
import java.time.*;
import java.time.temporal.ChronoUnit;

public class FindDifferenceBetweenDates {

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

		/*Syntaxes for creating specific date
		 * public satic void LocalDate.of(int year, int month, int dayOfMonth)
		 * or
		 * public static void LocalDate.of(int year, Month month, int dayOfMonth)
		 * Month.month is enum
		 */
		//now creating dates
		LocalDate date1 = LocalDate.of(2016,  7, 22);
		LocalDate date2 = LocalDate.of(2016,  7, 25);

		System.out.println("date1: " + date1);
		System.out.println("date2: " + date2);
		//finding days between the given dates using Java8 Date and TimeAPI
		long days = ChronoUnit.DAYS.between(date1, date2);
		System.out.println("Days between the given dates : "+days);
	}

}
```

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

```
3. import java.util.Scanner;
4.
5. public class DivdedByTwo {
6.
7.       //Using Bitwise operators << left shift 1( to multiply  with 2),
   >> right shift 1(to divde with 2)
8.       public static void main(String[] args){
9.             Scanner scanner = new Scanner(System.in);
10.                  System.out.println("Enter a number :");
11.                  int num = scanner.nextInt();
12.                  int quotient = num >> 1;
13.                  System.out.println("The Quotient of "+num+ "with 2
   with out using operator :"+quotient);
14.                  scanner.close();
15.
16.      }
17. }
```

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

```
import java.util.Scanner;

public class multiplyWithoutOperator {
      public static void main(String[] args){
      //Using Bitwise operators << left shift 1( to multiply  with 2), >>
right shift 1(to divde with 2)
      Scanner scanner = new Scanner(System.in);
      System.out.println("Enter a number :");
      int num = scanner.nextInt();
      int product = num << 1;
      System.out.println("The product of "+num+ " with 2 with out using
operator :"+product);
      scanner.close();
            }

      }
```

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

In Java we can't use pass -by-reference methods, because Java is Pass-by-Value language. This means that a copy of the variableis made and the method receives that copy. Assignments made in that method do not effect the caller.

5. **How to make a list immutable?**

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
public class ImmutableList {
    public static void main(String[] args) {
            // create array list
            List<Character> list = new ArrayList<Character>();
            // populate the list
            list.add('X');
            list.add('Y');
            System.out.println("Initial list: "+ list);
            // make the list unmodifiable
            List<Character> immutablelist = Collections.unmodifiableList(list);
            // try to modify the list
            immutablelist.add('Z');
            }
}
```

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

```
class LinkedList {
    static Node head;
    static class Node {
        int data;
        Node next;
        Node(int d) {
            data = d;
            next = null;
        }
    }
    /* Function to reverse the linked list */
    Node reverse(Node node) {
        Node prev = null;
        Node current = node;
        Node next = null;
        while (current != null) {
            next = current.next;
            current.next = prev;
            prev = current;
            current = next;
        }
        node = prev;
        return node;
    }
```

```
    // prints content of double linked list
    void printList(Node node) {
        while (node != null) {
            System.out.print(node.data + " ");
            node = node.next;
        }
    }

    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        list.head = new Node(85);
        list.head.next = new Node(15);
        list.head.next.next = new Node(4);
        list.head.next.next.next = new Node(20);
        System.out.println("Original Linked list is :");
        list.printList(head);
        head = list.reverse(head);
        System.out.println("");
        System.out.println("Reversed linked list : ");
        list.printList(head);
    }
}
```

Time Complexity: O(n)

7. **Write a program to implement ArrayList and Linked list**

import java.util.Arrays;

```
class ArrayListCustom<E> {
 private static final int INITIAL_CAPACITY = 10;
 private Object elementData[]={};
 private int size = 0;
 /**
 * constructor.
 */
 public ArrayListCustom() {
  elementData = new Object[INITIAL_CAPACITY];
 }
 /**
  * method adds elements in ArrayListCustom.
  */
 public void add(E e) {
  if (size == elementData.length) {
    ensureCapacity(); //increase current capacity of list, make it double.
  }
  elementData[size++] = e;
 }
```

```
 /**
  * method returns element on specific index.
  */
 @SuppressWarnings("unchecked")
 public E get(int index) {
   if ( index <0 || index>= size) {  //if index is negative or greater than size of size, we throw
Exception.
     throw new IndexOutOfBoundsException("Index: " + index + ", Size " + index);
   }
   return (E) elementData[index]; //return value on index.
 }



 /**
  * method returns removedElement on specific index.
  * else it throws IndexOutOfBoundException if index is negative or greater than size of size.
  */
 public Object remove(int index) {
   if ( index <0 || index>= size) {  //if index is negative or greater than size of size, we throw
Exception.
     throw new IndexOutOfBoundsException("Index: " + index + ", Size " + index);
   }

   Object removedElement=elementData[index];
   for(int i=index;i<size;i++){
     elementData[i]=elementData[i+1];
   }
   size--;   //reduce size of ArrayListCustom after removal of element.
```

```
    return removedElement;
  }



  /**
   * method increases capacity of list by making it double.
   */
  private void ensureCapacity() {
    int newIncreasedCapacity = elementData.length * 2;
    elementData = Arrays.copyOf(elementData, newIncreasedCapacity);
  }

  /**
   * method displays all the elements in list.
   */
  public void display() {
     System.out.print("Displaying list : ");
     for(int i=0;i<size;i++){
          System.out.print(elementData[i]+" ");
     }
  }

}

/**
 * Main class to test ArrayListCustom functionality.
 */
public class ArrayListCustomApp {
```

```
    public static void main(String...a) {
        ArrayListCustom<Integer> list = new ArrayListCustom<Integer>();
   list.add(1);
   list.add(2);
   list.add(3);
   list.add(4);
   list.add(1);
   list.add(2);

   list.display();
   System.out.println("\nelement at index "+1+" = "+list.get(1));
   System.out.println("element removed from index "+1+" = "+list.remove(1));

   System.out.println("\nlet's display list again after removal at index 1");

   list.display();

   //list.remove(11);  //will throw IndexOutOfBoundsException, because there is no element to remove on index 11.
   //list.get(11);   //will throw IndexOutOfBoundsException, because there is no element to get on index 11.

    }

}
/*Output
Displaying list : 1 2 3 4 1 2
element at index 1 = 2
```

element removed from index 1 = 2

let's display list again after removal at index 1

Displaying list : 1 3 4 1 2

*/

//Implementing LinkedList

```
public class SinglyLinkedListImpl<T> {

    private Node<T> head;
    private Node<T> tail;

    public void add(T element){

        Node<T> nd = new Node<T>();
        nd.setValue(element);
        System.out.println("Adding: "+element);
        /**
         * check if the list is empty
         */
        if(head == null){
            //since there is only one element, both head and
            //tail points to the same object.
            head = nd;
            tail = nd;
        } else {
            //set current tail next link to new node
            tail.setNextRef(nd);
            //set tail as newly created node
            tail = nd;
        }
    }

    public void addAfter(T element, T after){

        Node<T> tmp = head;
        Node<T> refNode = null;
        System.out.println("Traversing to all nodes..");
        /**
         * Traverse till given element
         */
        while(true){
            if(tmp == null){
                break;
            }
            if(tmp.compareTo(after) == 0){
                //found the target node, add after this node
```

```
                refNode = tmp;
                break;
            }
            tmp = tmp.getNextRef();
        }
        if(refNode != null){
            //add element after the target node
            Node<T> nd = new Node<T>();
            nd.setValue(element);
            nd.setNextRef(tmp.getNextRef());
            if(tmp == tail){
                tail = nd;
            }
            tmp.setNextRef(nd);


        } else {
            System.out.println("Unable to find the given element...");
        }
    }


    public void deleteFront(){

        if(head == null){
            System.out.println("Underflow...");
        }
        Node<T> tmp = head;
        head = tmp.getNextRef();
        if(head == null){
            tail = null;
        }
        System.out.println("Deleted: "+tmp.getValue());
    }


    public void deleteAfter(T after){

        Node<T> tmp = head;
        Node<T> refNode = null;
        System.out.println("Traversing to all nodes..");
        /**
         * Traverse till given element
         */
        while(true){
            if(tmp == null){
                break;
            }
            if(tmp.compareTo(after) == 0){
                //found the target node, add after this node
                refNode = tmp;
                break;
            }
            tmp = tmp.getNextRef();
        }
        if(refNode != null){
```

```
            tmp = refNode.getNextRef();
            refNode.setNextRef(tmp.getNextRef());
            if(refNode.getNextRef() == null){
                tail = refNode;
            }
            System.out.println("Deleted: "+tmp.getValue());
        } else {
            System.out.println("Unable to find the given element...");
        }
    }


    public void traverse(){

        Node<T> tmp = head;
        while(true){
            if(tmp == null){
                break;
            }
            System.out.println(tmp.getValue());
            tmp = tmp.getNextRef();
        }
    }


    public static void main(String a[]){
        SinglyLinkedListImpl<Integer> sl
= new SinglyLinkedListImpl<Integer>();
        sl.add(3);
        sl.add(32);
        sl.add(54);
        sl.add(89);
        sl.addAfter(76, 54);
        sl.deleteFront();
        sl.deleteAfter(76);
        sl.traverse();


    }
}


class Node<T> implements Comparable<T> {


    private T value;
    private Node<T> nextRef;


    public T getValue() {
        return value;
    }
    public void setValue(T value) {
        this.value = value;
    }
    public Node<T> getNextRef() {
```

```
        return nextRef;
    }
    public void setNextRef(Node<T> ref) {
        this.nextRef = ref;
    }
    @Override
    public int compareTo(T arg) {
        if(arg == this.value){
            return 0;
        } else {
            return 1;
        }
    }
}
```

8.Write a program for Insertion Sort in java.

```
public class InsertionOrder {
       public static void main(String[] args) {

            int[] input = { 4, 2, 9, 6, 23, 12, 34, 0, 1 };
            insertionSort(input);
         }

        private static void printNumbers(int[] input) {

            for (int i = 0; i < input.length; i++) {
                System.out.print(input[i] + ", ");
            }
            System.out.println("\n");
         }

         public static void insertionSort(int array[]) {
            int n = array.length;
            for (int j = 1; j < n; j++) {
                int key = array[j];
                int i = j-1;
                while ( (i > -1) && ( array [i] > key ) ) {
                    array [i+1] = array [i];
                    i--;
                }
                array[i+1] = key;
                printNumbers(array);
            }
         }
           }
```

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

```
import java.io.BufferedReader;
	import java.io.DataInputStream;
	import java.io.FileInputStream;
	import java.io.FileNotFoundException;
	import java.io.IOException;
	import java.io.InputStreamReader;
	import java.util.ArrayList;
	import java.util.List;
	import java.util.StringTokenizer;

	public class FindDistinctWords {
	    public List<String> getDistinctWordList(String fileName){

	        FileInputStream fis = null;
	        DataInputStream dis = null;
	        BufferedReader br = null;
	        List<String> wordList = new ArrayList<String>();
	        try {
	            fis = new FileInputStream(fileName);
	            dis = new DataInputStream(fis);
	            br = new BufferedReader(new InputStreamReader(dis));
	            String line = null;
	            while((line = br.readLine()) != null){
	                StringTokenizer st = new StringTokenizer(line, "
,.;:\"");

	                while(st.hasMoreTokens()){
	                    String tmp = st.nextToken().toLowerCase();
	                    if(!wordList.contains(tmp)){
	                        wordList.add(tmp);
	                    }
	                }
	            }
	        } catch (FileNotFoundException e) {
	            e.printStackTrace();
	        } catch (IOException e) {
	            e.printStackTrace();
	        } finally{
	            try{if(br != null) br.close();}catch(Exception ex){}
	        }
	        return wordList;
	    }
	    public static void main(String a[]){

	      FindDistinctWords distFw = new FindDistinctWords();
	        List<String> wordList =
distFw.getDistinctWordList("C:/Users/KOMMURI/Documents/canvas/JavaTest/ReadFi
le.txt");
	        for(String str:wordList){
	            System.out.println(str);
	        }
	    }
	}
```

10. **Find longest substring without repeating characters.**

```
import java.util.HashSet;
import java.util.Set;
public class LongestSubstring {
      private Set<String> subStrList = new HashSet<String>();
    private int finalSubStrSize = 0;

    public Set<String> getLongestSubstr(String input){
        //reset instance variables
        subStrList.clear();
        finalSubStrSize = 0;
        // have a boolean flag on each character ascii value
        boolean[] flag = new boolean[256];
        int j = 0;
        char[] inputCharArr = input.toCharArray();
        for (int i = 0; i < inputCharArr.length; i++) {
            char c = inputCharArr[i];
            if (flag[c]) {
                extractSubString(inputCharArr,j,i);
                for (int k = j; k < i; k++) {
                    if (inputCharArr[k] == c) {
                        j = k + 1;
                        break;
                    }
                    flag[inputCharArr[k]] = false;
                }
            } else {
                flag[c] = true;
            }
        }
        extractSubString(inputCharArr,j,inputCharArr.length);
        return subStrList;
    }

    private String extractSubString(char[] inputArr, int start, int end){

        StringBuilder sb = new StringBuilder();
        for(int i=start;i<end;i++){
            sb.append(inputArr[i]);
        }
        String subStr = sb.toString();
        if(subStr.length() > finalSubStrSize){
            finalSubStrSize = subStr.length();
            subStrList.clear();
            subStrList.add(subStr);
        } else if(subStr.length() == finalSubStrSize){
            subStrList.add(subStr);
        }

        return sb.toString();
    }

    public static void main(String a[]){
      LongestSubstring mls = new LongestSubstring();
        System.out.println(mls.getLongestSubstr("application"));
        System.out.println(mls.getLongestSubstr("java_language_is_sweet"));
```

```
        System.out.println(mls.getLongestSubstr("java_java_java_java"));
        System.out.println(mls.getLongestSubstr("abcabcbb"));
    }

}
```

11. **Write a program to remove duplicates from sorted array**

```
public class RemoveDuplcates {
       public static int[] removeDuplicates(int[] input){

             int j = 0;
             int i = 1;
             //return if the array length is less than 2
             if(input.length < 2){
                 return input;
             }
             while(i < input.length){
                 if(input[i] == input[j]){
                     i++;
                 }else{
                     input[++j] = input[i++];
                 }
             }
             int[] output = new int[j+1];
             for(int k=0; k<output.length; k++){
                 output[k] = input[k];
             }

             return output;
          }

          public static void main(String a[]){
             int[] input1 = {2,3,6,6,8,9,10,10,10,12,12};
             int[] output = removeDuplicates(input1);
             for(int i:output){
                 System.out.print(i+" ");
             }
          }

}
```

12. **Write a program to print fibonacci series**

```
public class FibonacciRecurrsion {
      static int n1=0,n2=1,n3=0;
       static void printFibonacci(int count){
          if(count>0){
               n3 = n1 + n2;
               n1 = n2;
               n2 = n3;
               System.out.print(" "+n3);
               printFibonacci(count-1);
           }
       }
       public static void main(String args[]){
        int count=10;
        System.out.print(n1+" "+n2);//printing 0 and 1
```

```
        printFibonacci(count-2);//n-2 because 2 numbers are already printed
       }
      }
```

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

```
import java.util.HashMap;
import java.util.Map;
import java.util.Set;

public class FindDuplicates {
      public void findDuplicateChars(String str){

        Map<Character, Integer> dupMap = new HashMap<Character, Integer>();
        char[] chrs = str.toCharArray();
        for(Character ch:chrs){
            if(dupMap.containsKey(ch)){
                dupMap.put(ch, dupMap.get(ch)+1);
            } else {
                dupMap.put(ch, 1);
            }
        }
        Set<Character> keys = dupMap.keySet();
        for(Character ch:keys){
            if(dupMap.get(ch) > 1){
                System.out.println(ch+"--->"+dupMap.get(ch));
            }
        }
    }

    public static void main(String a[]){
      FindDuplicates dcs = new FindDuplicates();
        dcs.findDuplicateChars("Hello World");
    }

}
```

14. Write a program to create deadlock between two threads

```
public class DeadLock {
      String str1 = "Java";
    String str2 = "UNIX";

    Thread trd1 = new Thread("My Thread 1"){
        public void run(){
            while(true){
                synchronized(str1){
                    synchronized(str2){
                        System.out.println(str1 + str2);
                    }
                }
            }
        }
    };

    Thread trd2 = new Thread("My Thread 2"){
```

```
        public void run(){
            while(true){
                synchronized(str2){
                    synchronized(str1){
                        System.out.println(str2 + str1);
                    }
                }
            }
        }
    };

    public static void main(String a[]){
       DeadLock mdl = new DeadLock();
        mdl.trd1.start();
        mdl.trd2.start();
    }
}
```

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

```
public class FindMiddleIndex {
      public static int findMiddleIndex(int[] numbers) throws Exception {

        int endIndex = numbers.length - 1;
        int startIndex = 0;
        int sumLeft = 0;
        int sumRight = 0;
        while (true) {
            if (sumLeft > sumRight) {
                sumRight += numbers[endIndex--];
            } else {
                sumLeft += numbers[startIndex++];
            }
            if (startIndex > endIndex) {
                if (sumLeft == sumRight) {
                    break;
                } else {
                    throw new Exception(
                            "Please pass proper array to match the
requirement");
                }
            }
        }
        return endIndex;
    }

    public static void main(String a[]) {
        int[] num = { 2, 4, 4, 5, 4, 1 };
        try {
            System.out.println("Starting from index 0, adding numbers till
index "
                               + findMiddleIndex(num) + " and");
            System.out.println("adding rest of the numbers can be equal");
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
```

```
    }

}
```

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

```
public class ArmStrongOrNot {
      public boolean isArmstrongNumber(int number){

        int tmp = number;
        int noOfDigits = String.valueOf(number).length();
        int sum = 0;
        int div = 0;
        while(tmp > 0)
        {
            div = tmp % 10;
            int temp = 1;
            for(int i=0;i<noOfDigits;i++){
                temp *= div;
            }
            sum += temp;
            tmp = tmp/10;
        }
        if(number == sum) {
            return true;
        } else {
            return false;
        }
    }

    public static void main(String a[]){
      ArmStrongOrNot man = new ArmStrongOrNot();
        System.out.println("Is 371 Armstrong number?
"+man.isArmstrongNumber(371));
        System.out.println("Is 523 Armstrong number?
"+man.isArmstrongNumber(523));
        System.out.println("Is 153 Armstrong number?
"+man.isArmstrongNumber(153));
    }

}
```