**1st Round :**

1. **Array has 1000 elements between 1-9. Count the occurrence of each digit in the array. Try not using Collection APIs**

package Gopi;

public class NumberOfTimes {

public static void main(String[] args) {

// TODO Auto-generated method stub

int[] arr = { 1, 2, 3, 1, 4, 2, 5, 3, 1, 6, 6, 7, 8, 9, 4, 5, 6, 1, 2,

3, 4, 5, 6, 7, 8, 9 };

int[] incr=new int[9];

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

switch (arr[i]) {

case 1:

incr[0]++;

break;

case 2:

incr[1]++;

break;

case 3:

incr[2]++;

break;

case 4:

incr[3]++;

break;

case 5:

incr[4]++;

break;

case 6:

incr[5]++;

break;

case 7:

incr[6]++;

break;

case 8:

incr[7]++;;

break;

case 9:

incr[8]++;

break;

}

}

for(int j=0;j<9;j++)

{

System.out.println(j+1+"repeated "+incr[j]+" times");

}

}

}

1. **Write a program to calculate the factorial of a given number(both by normal method and using recursion)**
2. import java.util.Scanner;

class Factorial

{

public static void main(String args[])

{

int n, c, fact = 1;

System.out.println("Enter an integer to calculate it's factorial");

Scanner in = new Scanner(System.in);

n = in.nextInt();

if ( n < 0 )

System.out.println("Number should be non-negative.");

else

{

for ( c = 1 ; c <= n ; c++ )

fact = fact\*c;

System.out.println("Factorial of "+n+" is = "+fact);

}

}

}

1. **package** Gopi;

**import** java.util.Scanner;

**public** **class** FactorialRecursion {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("enter a number");

**int** num=sc.nextInt();

System.***out***.println("factorial of "+num +" is "+*fact*(num));

}

**private** **static** **int** fact(**int** num) {

// **TODO** Auto-generated method stub

**if**(num<=1)

**return** 1;

**else** **return** num\**fact*(num-1);

}

}

1. **Write a program to print Fibonacci series.**
2. package com.java2novice.algos;

public class MyFibonacci {

public static void main(String a[]){

int febCount = 15;

int[] feb = new int[febCount];

feb[0] = 0;

feb[1] = 1;

for(int i=2; i < febCount; i++){

feb[i] = feb[i-1] + feb[i-2];

}

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

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

}

}

}

1. package Gopi;

import java.util.Scanner;

public class Fibonaci {

public static void main(String[] args) {

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

System.out.println("Enter Range");

int limit=sc.nextInt();

int i=0,j=1,k=i+j;

System.out.println(i);

System.out.println(j);

System.out.println(k);

while(k<=limit)

{

i=j;

j=k;

k=i+j;

if(k<=limit)

System.out.println(k);

}

}

}

1. **package** Gopi;

**import** java.util.Scanner;

**public** **class** FibonaciUsingRecursion {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner sc=**new** Scanner(System.***in***);

**int** index=0;

System.***out***.println("enter range");

**int** range=sc.nextInt();

**while** (index<=range)

{

System.***out***.println(*fibonaci*(index));

index++;

}

}

**private** **static** **int** fibonaci(**int** range) {

// **TODO** Auto-generated method stub

**if**(range==0)

**return** 0;

**else** **if**(range==1)

**return** 1;

**else**

**return** *fibonaci*(range-1)+*fibonaci*(range-2);

}

}

1. **Write a program to check of whether a string is Palindrome or not.**
2. import java.util.\*;

class Palindrome

{

public static void main(String args[])

{

String original, reverse = "";

Scanner in = new Scanner(System.in);

System.out.println("Enter a string to check if it is a palindrome");

original = in.nextLine();

int length = original.length();

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

reverse = reverse + original.charAt(i);

if (original.equals(reverse))

System.out.println("Entered string is a palindrome.");

else

System.out.println("Entered string is not a palindrome.");

}

}

1. **package** Gopi;

**import** java.util.Scanner;

**public** **class** StringPalendrome {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Scanner sc = **new** Scanner(System.***in***);

System.***out***.println("enter a string");

String str = sc.nextLine();

StringBuffer sb = **new** StringBuffer(str);

**if** (str.equals((sb.toString())))

System.***out***.println("palendrome");

**else**

System.***out***.println("not a palendrome");

}

}

**2nd Round :**

1. **Write a query to display duplicate records.**

SELECT \* FROM table1

WHERE (columnname,columnname) IN (SELECT columnname,columnname FROM table1

GROUP BY columnname,columnname

HAVING COUNT(\*) > 1)

1. **Given a linkedlist. delete the middle node . u have only access to node “c”.**

**o/p:: should be a->b->d->e .**

public boolean delete(Node ln)

{

Node ln1=ln;

Int length=ln.length/2;

// for this we have to find the middle node and delete logically i.e remove the link with next //node.

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

{

ln=ln.next;

}

If(ln==null || ln.next==null)

return false;

Node next=ln.next;

ln.data=next.data;

ln.next=next.next;

return true;

}

1. **Tell me about yourself.**
2. **Explain the program written in 1st round.**
3. **Difference between StringBuffer, StringBuilder.**
4. **Reverse the elements in the string.**
5. **Reverse the LinkedList.**
6. **Questions on Queue and Stack logic.**
7. **Questions on Joins.**
8. **SQL : Find out names which are starting with ‘S’.**
9. **Time Complexity for different Sorting Techniques.**
10. **Which is best Sorting Technique.**
11. **Logic to find the missing element in an array.**
12. **Questions on Unix.**
13. **Question on Project.**

**3rd Round :**

1. **Best case sorting when 100 elements need to be sorted in a memory space of 10 elements.**
2. **Question on Project.**