**Program 1: To check if a number entered by the user is a Smith Number or not.**

import java.util.\*;

class SmithNumber

{

Scanner s=new Scanner(System.in);

//function for finding sum of digits

int sumDig(int n)

{

int s=0;

while(n>0)

{

s=s+n%10;

n=n/10;

}

return s;

}

//function for generating prime factors and finding their sum

int sumPrimeFact(int n)

{

int i=2, sum=0;

while(n>1)

{

if(n%i==0)

{

sum=sum+sumDig(i); //Here 'i' is the prime factor of 'n' and we are finding its sum

n=n/i;

}

else

i++;

}

return sum;

}

void main()

{

SmithNumber ob=new SmithNumber();

System.out.print("Enter a Number : ");

int n=s.nextInt();

int a=ob.sumDig(n);// finding sum of digit

int b=ob.sumPrimeFact(n); //finding sum of prime factors

System.out.println("Sum of Digit = "+a);

System.out.println("Sum of Prime Factor = "+b);

if(a==b)

System.out.print("It is a Smith Number");

else

System.out.print("It is Not a Smith Number");

}

}

**Program 2: To check if a number entered by the user is an Automorphic Number**

import java.util.\*;

class Automorphic

{

void main()

{

Scanner s=new Scanner(System.in);

System.out.print("Enter a Number : ");

int n = s.nextInt();

int sq = n\*n;

int c = 0, copy = n;

// While loop for counting the number of digits in the number

while(copy > 0)

{

c++;

copy = copy/10;

}

/\* Finding the end digits of the square.

\* If the number has 2 digits, then we need to find last 2 digits of square

\*/

int end = sq % (int)Math.pow(10,c);

if(n == end) // If the square ends with the number then it is Automorphic

System.out.print(n+" is an Automorphic Number.");

else

System.out.print(n+" is not an Automorphic Number.");

}

}

**Program 3: To check if a number entered by the user is a Kaprekar Number.**

import java.util.\*;

class KaprekarNumber

{

void main()

{

Scanner sc=new Scanner(System.in);

System.out.print("Enter a Number : ");

int n = sc.nextInt() ; //Inputting the number

int sq = n\*n;

String s = Integer.toString(sq); //converting the square into a String

if(sq<=9)

s = "0"+s; //Adding a zero in the beginning if the square is of single digit

int l = s.length(); //finding the length (i.e. no. of digits in the square).

int mid = l/2; //finding the middle point

String left=s.substring(0,mid); //extracting the left digits from the square

String right=s.substring(mid); //extracting the right digits from the square

int x = Integer.parseInt(left); //converting the left String into Integer

int y = Integer.parseInt(right); //converting the right String into Integer

//if sum of left and right numbers is equal to the original number then it is a Kaprekar number

if(x+y == n)

System.out.println(n+" is a Kaprekar Number");

else

System.out.println(n+" is Not a Kaprekar Number");

}

}

**Program 4: A simple encryption system uses a shifting process to hide a message. The value of the shift can be in the range 1 to 26. For example a shift of 7 means that A = U, B = V, C = W, etc., *i.e.,***

**Text: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z**

**Code: U V W X Y Z A B C D E F G H I J K L M N O P Q R S T**

**First an extra space is added to the end of the string. To make things a little more difficult, spaces within the original text are replaced with QQ before the text is encrypted. Double Q (QQ) was selected because no English word ends in Q or contains QQ.**

**Additionally the coded message is printed in blocks of six characters separated by spaces. The last block might not contain six characters. Write a program that takes the coded text (less than 100 characters), the shift value and prints the decoded original text. Your program must reject any non-valid value for shift and display an error message "INVALID SHIFT VALUE." Assume all characters are upper case.**

import java.util.\*;

public class Decode\_ISC2003

{

void main(String s, int shift)

{

int l = s.length();

s = s.toUpperCase();

s = s + " "; // adding a space at the end

if(l>=100)

System.out.println("Invalid Length of Coded Text");

else

{

System.out.print("The Shift Value is:"+shift);

if(shift<1 || shift>26)

System.out.println(" Invalid Shift Value ");

else

{

int a, b;

String dec=""; //new String for storing the decoded text

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

{

/\* Below we are adding shift value to the characters

\* if ch1 = 'A' and shift = 7,

\* then ch1 + shift - 1 will give us: 'A'+7-1 = 65+7-1 = 71

\* which is the ASCII value of 'G'

\*/

a = s.charAt(i) + shift - 1;

b = s.charAt(i+1) + shift - 1;

/\* If the currrent character and the next character are both 'Q' then we have a 'space'

\* hence the ASCII value should be 32

\*/

if((char)a == 'Q' && (char)b == 'Q')

{

a = 32;

i++;

}

/\* If ASCII value after adding the shift becomes more than 90,

\* then we subtract 26 from it, to make it circular,

\* eg. 'U'+7-1 = 85+7-1 = 91, but we want 'A' whose ASCII value is 65

\* so 91-26 will give us 65

\*/

if(a>90)

a = a - 26;

if(s.charAt(i) != ' ')

dec = dec + (char)a;

}

System.out.println("Decoded Text : "+dec);

}

}

}

**Program 5: To sort an array using Bubble Sort technique.**

class BubbleSort

{

void main(int a[])

{

int backup=0;//variable used for the switching purpose

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

{

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

{

if(a[j]>a[j+1])

{

backup=a[j];//to store the value of the greater element in 'backup'

a[j]=a[j+1];

a[j+1]=backup;

}

}

}

for(int k=0;k<a.length;k++)//loop for printing the elements

{

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

}

}

}

**Program 5: To sort an array using Selection Sort technique.**

class SelectionSort

{

void main(int a[])

{

int smallest=0;//to store smallest digit in the array

int pos=0;//to store the position of the smallest variable

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

{

smallest=a[i];

pos=i;

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

{

if(a[j]>smallest)

{

smallest=a[j];

pos=j;

}

}

a[pos]=a[i];

a[i]=smallest;

System.out.println("The Sorted Array is:");

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

}

}

}

**Program 6: To search for a number in an array using Binary Search technique.**

class BinarySearch

{

void main(int a[],int num)

{

int f=0; int l=a.length-1;

int m=(f+l)/2;

boolean flag=false;

while(f<=l)

{

if(a[m]>num)

{

l=m-1;

m=(f+l)/2;

}

else if(a[m]<num)

{

f=m+1;

m=(f+l)/2;

}

else if(a[m]==num)

{

flag=true;

break;

}

}

if(flag==true)

System.out.println("The number "+num+" is present");

}

}

**Program 7: Write a program to declare a square matrix A[ ] [ ] of order (M x M) where ‘M’ is the number of rows and the number of columns such that M must be greater than 2 and less than 20. Allow  the user to input integers into this matrix. Display appropriate error message for an invalid input. Perform the following tasks:**

**a) Display the input matrix**

**b) Create a mirror image matrix**

**c) Display the mirror image matrix**

**Test your program with the sample data and some random data:**

**Example 1**

INPUT : M = 3  
4       16      12  
8        2       14  
4        1        3

OUTPUT :

ORIGINAL MATRIX

4       16       12  
8         2       14  
4         1        3

MIRROR IMAGE MATRIX

12      16      4  
14       2      8  
3         1      6

**Example 2**

INPUT : M = 22

OUTPUT : SIZE OUT OF RANGE

import java.util.\*;

class MatrixMirrorImage

{

void main()

{

Scanner sc=new Scanner(System.in);

System.out.println("Size of Matrix");

int m=sc.nextInt();

if(m>2 && m<20) //checking given condition

{

int A[][]=new int[m][m];

System.out.println("Enter the elements of the Matrix : ");

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

{

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

{

A[i][j]=sc.nextInt();

}

}

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

System.out.println("The original matrix:");

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

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

{

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

{

System.out.print(A[i][j]+"\t");

}

System.out.println();

}

// creating the Image Matrix

int B[][]=new int[m][m];

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

{

int k=0;

for(int j=m-1;j>=0;j--)

{

B[i][k]=A[i][j];

k++;

}

}

//Printing both the Matrix

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

System.out.println("The Mirror Image:");

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

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

{

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

{

System.out.print(B[i][j]+"\t");

}

System.out.println();

}

}

else

System.out.println("Output : Size Out Of Range");

}

}

**Program 8: Stack is a linear data structure which enables the user to add and remove elements from the top end only, using the concept of LIFO (Last In First Out).**

**To implement a Stack:**

import java.util.\*;

class Stack

{

int top=0;

int f=0;

int a[];

void push()

{

Scanner s=new Scanner(System.in);

System.out.println("Enter number to be pushed:");

int b=s.nextInt();

if(f==top)

{

System.out.println("Stack is at its maximum height.");

return;

}

f++;

a[f-1]=b;

System.out.println(b+" has been pushed.");

return;

}

void pop()

{

if(f<=0)

{

System.out.println("All numbers have been popped.");

return;

}

System.out.println(a[f-1]+" has been popped.");

f--;

return;

}

void main()

{

Scanner s=new Scanner(System.in);

System.out.println("Enter the height of the stack:");

top=s.nextInt();

a=new int[top];

int v=0;

while(v!=3)

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter: 1.To push 2.To pop 3.To Exit");

v=sc.nextInt();

if(v==1)

{

push();

}

if(v==2)

{

pop();

}

if(v!=1&&v!=2&&v!=3)

{

System.out.println("Invalid Option.");

}

}

System.out.println("Program Terminated");

}

}

**Program 9: Queue is a linear data structure which enables the user to add elements from the rear end and remove elements from the front end only, using the concept of FIFO (First In First Out).**

**To implement a Queue:**

import java.util.\*;

class Queue

{

int f=0;

int length=0;

int rear=0;

int front=0;

int v=0;

int a[];

void push()

{

Scanner s=new Scanner(System.in);

System.out.println("Enter number to be pushed:");

int b=s.nextInt();

if(rear==length)

{

System.out.println("Queue has reached maximum length.");

return;

}

a[rear]=b;

rear++;

f++;

System.out.println(b+" has been pushed.");

return;

}

void pop()

{

if(f==0)

{

System.out.println("All members in queue have been popped.");

return;

}

System.out.println(a[front]+" has been popped.");

front++;

f--;

if(front==rear&&rear==length)

{

System.out.println("Queue has reached maximum length and all members have been popped.");

v=3;

return;

}

return;

}

void main()

{

Scanner s=new Scanner(System.in);

System.out.println("Enter the length of the queue:");

length=s.nextInt();

a=new int[length];

while(v!=3)

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter: 1.To push 2.To pop 3.To Exit");

v=sc.nextInt();

if(v==1)

{

push();

}

if(v==2)

{

pop();

}

if(v!=1&&v!=2&&v!=3)

{

System.out.println("Invalid Option.");

}

}

System.out.println("Program Terminated");

}

}

**Program 10: The Tower of Hanoi is a**[**mathematical game**](https://en.wikipedia.org/wiki/Mathematical_game)**or**[**puzzle**](https://en.wikipedia.org/wiki/Puzzle)**. It consists of three rods, and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a**[**conical**](https://en.wikipedia.org/wiki/Cone) **shape.**

**The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:**

1. **Only one disk can be moved at a time.**
2. **Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.**
3. **No disk may be placed on top of a smaller disk.**

**Code a program that prints the steps involved to complete the game according to the number disks involved.**

class TowerofHanoiRecursion

{

void main(int n)

{

char s='L',i='C',d='R';

move(n,s,i,d);

}

void move(int nn,char ss,char ii,char dd)

{

if(nn!=0)

{

move(nn-1,ss,dd,ii);

System.out.println("Move Disc "+nn+" from "+ss+" to "+dd);

move(nn-1,ii,ss,dd);

}

}

}

**Program 11: A linked list is a linear data structure where each element is a separate object. Each element (called a node) of a list is comprising of two items - the data and a reference to the next node. The last node has a reference to null.**

**Write a program that implements the different functions of a linked list.**

**Class Node:**

class Node

{

int data;

Node link;

Node()

{

data=0;

link=null;

}

Node(int d,Node n)

{

data=d;

link=n;

}

int getData()

{

return data;

}

void setData(int a)

{

data=a;

}

Node getLink()

{

return link;

}

void setLink(Node n)

{

link=n;

}

}

**Main Linked List:**

import java.util.\*;

class LinkedListComplete extends Node

{

Node start;

void add()//method to add nodes to the linked list

{

Scanner s=new Scanner(System.in);

System.out.println("Enter number to be added:");

int a=s.nextInt();

System.out.println();

Node nnode=new Node();

nnode.setData(a);

if(start==null)

{

start=nnode;

return;

}

int v=start.getData();

if(a<v)

{

nnode.setLink(start);

start=nnode;

return;

}

Node temp=start;

Node tempn=temp.getLink();

while(tempn!=null)

{

int x=temp.getData();

int y=tempn.getData();

if(a>x&&a<=y)

{

temp.setLink(nnode);

nnode.setLink(tempn);

return;

}

else

{

temp=temp.getLink();

tempn=tempn.getLink();

}

}

temp.setLink(nnode);

}

void display(Node temp)//method to display the nodes

{

System.out.println();

if(temp==null)

{

return;

}

else

{

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

System.out.println(temp.getData());

temp=temp.getLink();

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

display(temp);

}

}

void delete()//method to delete nodes from the linked list

{

if(start==null)

{

System.out.println("The Linked List is empty.");

return;

}

else

{

Scanner s=new Scanner(System.in);

System.out.println("Enter number to be deleted:");

int a=s.nextInt();

int z=0;

Node tempb=start;

Node temp=start;

while(temp!=null)

{

int b=temp.getData();

if(a==b)

{

if(z==0)

{

start=temp.getLink();

System.out.println(a+" has been deleted.");

return;

}

tempb.setLink(temp.getLink());

System.out.println(a+" has been deleted.");

return;

}

if(z==0)

{

z++;

}

else

{

tempb=tempb.getLink();

}

temp=temp.getLink();

}

System.out.println("Given number is not present in the list.");

}

}

int sum(Node temp)//to find sum of values of the nodes in the linked list

{

System.out.println();

if(temp==null)

{

return 0;

}

else

{

int ns=temp.getData();

temp=temp.getLink();

return ns+sum(temp);

}

}

int num(Node temp)//to print the number of nodes in the linked list

{

if(temp==null)

{

return 0;

}

else

{

temp=temp.getLink();

return 1+num(temp);

}

}

void main()//to access the particular method depending on the user’s needs

{

int v=0;

while(v!=6)

{

Scanner s=new Scanner(System.in);

System.out.println("Enter: 1.To Add 2.To Display 3.To Delete 4.To Print Sum 5.To Print No.of Nodes 6.To Exit");

v=s.nextInt();

if(v==1)

{

add();

}

if(v==2)

{

if(start==null)

{

System.out.println("The Linked List is empty.");

continue;

}

display(start);

}

if(v==3)

{

delete();

}

if(v==4)

{

if(start==null)

{

System.out.println("The Linked List is empty.");

continue;

}

int nsum=sum(start);

System.out.println("The sum of the nodes is: "+nsum);

}

if(v==5)

{

if(start==null)

{

System.out.println("There are 0 nodes in the Linked List");

continue;

}

int n=num(start);

System.out.println("There are "+n+" nodes int the Linked List");

}

if(v!=1&&v!=2&&v!=3&&v!=4&&v!=5&&v!=6)

{

System.out.println("Invalid Input");

}

System.out.println();

}

}

}

**Program 12: A class BinarySearchRecursion contains the admission numbers of 100 students. Some of the data members/ member functions are given below:**

**Class name: BinarySearchRecursion**

**Data member/instance variable:**

**Adno[ ]: Integer array to store admission numbers**

**Member functions/methods:**

**BinarySearchRecursion(): constructor to initialize the array elements.**

**void fillarray(): to accept elements of the array in ascending order.**

**int BinSearch(int l, int u, int v): to search for a particular admission number(v) using binary search and recursive technique and return 1 if found else return -1.**

**Specify the class Admission giving details of the constructor, void fillArrray() and int binSearch(int, int, int). Define the main() function to create an object and call the functions accordingly to enable task.**

import java.util.\*;

class BinarySearchRecursion

{

int Adno[]=new int[100];

Scanner sc = new Scanner(System.in);

BinarySearchRecursion() // Default constructor

{

for(int i=0; i<100; i++)

{

Adno[i]=0;

}

}

void fillArray()// Function to accept elements in ascending order

{

for(int i=0; i<100; i++)

{

System.out.print("Enter Admission no of student "+(i+1)+": ");

Adno[i] = sc.nextInt();

}

/\*Sorting the array in ascending order \*/

int temp=0;

for(int i=0; i<99; i++)

{

for(int j=i+1; j<100; j++)

{

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

{

temp = Adno[i];

Adno[i] = Adno[j];

Adno[j] = temp;

}

}

}

}

int binSearch(int l, int u, int v) // Recursive function implementing binary search

{

int mid = (l + u)/2;

if(u < l) // condition if the search is unsuccessful

{

return -1;

}

if(v==Adno[mid]) // condition if the search is successful

{

return 1;

}

else if(v>Adno[mid])

{

return binSearch(mid+1,u,v);

}

else

{

return binSearch(l,mid-1,v);

}

}

void main()

{

BinarySearchRecursion ob = new BinarySearchRecursion();

System.out.println("Enter Admission number in ascending order");

ob.fillArray();

System.out.print("Enter an Admission number to search : ");

int v = sc.nextInt();

int f = ob.binSearch(0,99,v);

if(f == 1)

{

System.out.println("Admission Number found");

}

else

{

System.out.println("Admission Number Not found");

}

}

}

**Program 13: A class Recursion has been defined to find the Fibonacci series upto a limit. Some of the members of the class are given below:**

**Class Name : Recursion**

**Data Members/instance variables : a, b, c, limit (all integers)**

**Member functions/methods :**

**Recursion() : constructor to assign a,b,c with appropriate values.  
void input() : to accept the limit of the series.  
int fib(int n) : to return the nth Fibonacci term using recursive technique.  
void genearate\_fibseries() : to generate the Fibonacci series upto the given limit.**

**Specify the class Recursion giving details of the constructor, int fib() , void generate\_fibseries(). You may assume other functions are written for you and you need not write the main function.**

import java.util.\*;

class Recursion

{

Scanner s=new Scanner(System.in);

int a,b,c,limit;

Recursion() //Constructor

{

a=0;

b=1;

c=0;

limit=0;

}

void input()

{

System.out.print("Enter the limit : ");

limit=s.nextInt();

}

int fib(int n) //Recursive function generating the 'nth' term of Fibonacci Series

{

if(n<=1)

return a;

else if(n==2)

return b;

else

return (fib(n-1)+fib(n-2));

}

void fibseries()

{

System.out.println("The Fibonacci Series is:");

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

{

c=fib(i);

System.out.print(c+" ");

}

}

void main()

{

Recursion ob=new Recursion();

ob.input();

ob.fibseries();

}

}

**Program 14: To find HCF of two numbers**

import java.util.\*;

class Hcf

    {

         void main()

        {

            Scanner s=new Scanner(System.in);

System.out.print("Enter the First no : ");

            int n1=s.nextInt();

 System.out.print("Enter the Second no : ");

            int n2=s.nextInt();

  int hcf=0;

           int min = Math.min(n1,n2);

            for(int i=min; i >= 1; i--)

            {

                if(n1%i == 0 && n2%i == 0)

                {

 hcf = i;

                    break;

}

            }

            System.out.print("\nThe hcf of "+n1+" and "+n2+" = "+hcf);

        }

    }

**Program 15: To find LCM of two numbers**

import java.util.\*;

class LCM

{

void main()

{

Scanner s=new Scanner(System.in);

int a,b,lcm=1;

System.out.print("Enter the 1st number : ");

a=s.nextInt();

System.out.print("Enter the 2nd number : ");

b=s.nextInt();

for(int i=a;i<=a\*b;i++) //Even if you start the for loop by 1, you will get the answer, but starting it from either the first or the second number reduces the number of times the for loop is executed.

{

if(i%a==0 && i%b==0) //Checking the first number which is divisible by both the numbers

{

lcm=i;

break;

}

}

System.out.println("L.C.M. = "+lcm);

}

}

**Program 16: Sorting a 2D Array**

import java.util.\*;

class Sort2DArray

{

void main()

{

Scanner s=new Scanner(System.in);

System.out.println("Enter the number of rows:");

int m=s.nextInt();

System.out.println("Enter the number of columns:");

int n=s.nextInt();

int a[][]=new int[m][n];

 /\* Inputting the 2D Array \*/

System.out.print("Enter the elements of array:");

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

{

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

{

a[i][j]=s.nextInt();

}

}

 /\* Printing the original 2D Array \*/

System.out.println("Original Array:");

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

{

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

{

System.out.print(a[i][j]);

}

System.out.println();

}

/\* Sorting the 2D Array \*/

int t=0;

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

{

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

{

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

{

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

{

if(a[i][j]>a[x][y])

{

t=a[x][y];

a[x][y]=a[i][j];

a[i][j]=t;

}

}

}

}

}

/\* Printing the sorted 2D Array \*/

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

{

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

{

System.out.print(a[i][j]);

}

System.out.println();

}

}

}

**Program 17: To read a file**

import java.util.Scanner;

import java.io.\*;

class File\_Read

{

void scan\_read()throws FileNotFoundException, IOException

{

File f = new File("E:\\2015-2016\\Class XII\\test84.txt");

BufferedReader br = new BufferedReader(new FileReader(f));

String s = br.readLine();

do

{ System.out.println(s);

s = br.readLine();

}while(s != null);

br.close();

System.out.println();

br = new BufferedReader(new FileReader(f));

Scanner scn = new Scanner(br);

while(scn.hasNextLine())

{

System.out.println(scn.nextLine());

}

System.out.println();

Scanner fr = new Scanner(f);

while(fr.hasNextLine())

{

System.out.println(fr.nextLine());

}

}

}

**Program 18: To write a file**

import java.io.\*;

class File\_Write

{

void write(int a) throws IOException

{

File f = new File("E:\\2015-2016\\Class XII\\test"+a+".txt");

PrintWriter pw = new PrintWriter((new FileWriter(f)));

a\*=2;

for(int i=1; i <= (a/2); i++)

{

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

{

pw.print("blah ");

}

for(int j=0; j < (a-2\*i); j++)

{

pw.print(" ");

}

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

{

pw.print("blah ");

}

pw.println();

}

pw.close();

}

}

**Program 19: A new advanced Operating System, incorporating the latest hi-tech features has been designed by Opera Computer Systems.**

**The task of generating copy protection codes to prevent software piracy has been entrusted to the Security Department.**

**The Security Department has decided to have codes containing a jumbled combination of alternate uppercase letters of the alphabet starting from A upto K (Namely among A,C,E,G,I,K). The code may or may not be in the consecutive series of alphabets.**

**Each code should not exceed 6 characters and there should be no repetition of characters. If it exceeds 6 Characters, display an appropriate error message.**

**Write a program to input a code and its length. At the first instance of an error display “Invalid!” stating the appropriate reason. In case of no error, display the message “ Valid!”**

**SAMPLE DATA**

**INPUT**

**N = 4**

**ABCE**

**OUTPUT**

**Invalid! Only alternate letters permitted!**

**INPUT**

**N =5**

**GEAIK**

**OUTPUT**

**valid!**

import java.util.\*;

class OperatingSystemCode

{

void main()

{

Scanner s=new Scanner(System.in);

System.out.println("Enter the code");

String code=s.nextLine();

System.out.println("Enter the length of the code");

int length=s.nextInt();

boolean flag=false;

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

{

char c=code.charAt(i);

if(code.length()!=length)

{

System.out.println("Invalid! String length not same as specified!");

flag=true;

break;

}

else if(length>6||code.length()>6)

{

System.out.println("Invalid! Length of the string should not exceed 6 characters!");

flag=true;

break;

}

else if(Character.isUpperCase(c)==false)

{

System.out.println("Invalid! Only upper case letters are permitted!");

flag=true;

break;

}

else if(c!='A'&&c!='C'&&c!='E'&&c!='G'&&c!='I'&&c!='K')

{

System.out.println("Invalid! Only alternate letters are permitted!");

flag=true;

break;

}

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

{

if(c==code.charAt(j)&&(i!=j))

{

System.out.println("Invalid! Repeated letters are not permitted!");

flag=true;

break;

}

}

if(flag==true)

{

break;

}

}

if(flag==false)

{

System.out.println("Valid!");

}

}

}

**Program 20: Write a program to calculate and print the corresponding day of the year (in the range 1 to 366 )**

**Example :**

**Input: Month number: 05**

**Day 03**

**Year 1996**

**Output: Corresponding day of the year is 124**

class Day\_of\_year

{

void main(int month,int day,int year)

{

boolean f=false;

if(month<=12)

{

if(month<=7)

{

if(month%2==0)

{

if(year%4==0)

{

if(day<=29)

{

f=true;

}

else

{

f=false;

}

}

else

{

if(month==2)

{

if(day<=28)

{

f=true;

}

else

{

f=false;

}

}

if(month!=2&&day<=30)

{

f=true;

}

}

}

else

{

if(month%2==0)

{

if(day<=31)

{

f=true;

}

else

{

f=false;

}

}

else

{

if(day>30)

{

f=false;

}

else

{

f=true;

}

}

}

}

else

{

f=false;

}

if(f==true)

{

System.out.println("VALID DATE");

int a=0;

if(year%4==0)

{

a=29;

}

else

{

a=28;

}

int sum=0;

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

{

if(i<8)

{

if(i==2)

{

sum=sum+a;

}

else

{

if(i%2!=0)

{

sum=sum+31;

}

else

{

sum=sum+30;

}

}

}

else

{

if(i%2==0)

{

sum=sum+31;

}

else

{

sum=sum+30;

}

}

}

sum=sum+day;

System.out.print(sum);

}

else

{

System.out.print("INVALID DATE");

}

}

}

}

**Program 21: Prime factorization of a number is the determination of the set of prime numbers which when multiplied together gives the original integer. To generate the prime factors of a number:**

class Prime\_factorisation

{

void main(int n)

{int temp=n;

while(n>1)

{

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

{

int c=0;

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

{

if(i%j==0)

{

c++;

}

}

if(c==2)

{

if(n%i==0)

{

System.out.print(i+" ");

n=n/i; break;

}

}

}

}

}

}

**Program 22: To print the frequency of every word found in a sentence.**

class SentenceWordFrequency

{

void main(String s)

{

s=s+" ";

int c=0;

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

{

if(s.charAt(i)==' ')

{

c++;

}

}

String a[]=new String[c];

c=0;

String n="";

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

{

if(s.charAt(j)!=' ')

{

n=n+s.charAt(j);

}

else

{

boolean f=true;

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

{

if(n.equalsIgnoreCase(a[k]))

{

f=false;

}

}

if(f==true)

{

a[c]=n;

c++;

}

n="";

}

}

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

{

if(a[l]!=null)

{

c=0;

for(int m=0;m<s.length();m++)

{

if(s.charAt(m)!=' ')

{

n=n+s.charAt(m);

}

else

{

if(n.equalsIgnoreCase(a[l]))

{

c++;

}

n="";

}

}

System.out.println("Frequency of '"+a[l]+"' is "+c);

}

}

}

}

**Program 23: Krishnamoorthy Number is a number which is equal to the sum of the factorials of all its digits.  
  
For example : 145 = 1! + 4! + 5! = 1 + 24 + 120 = 145**

**To check if a given number is a Krishnamoorthy Number or not.**

class KrishnamoorthyNumber

{

void main(int n)

{

int a=n;

int sum=0;

while(n>0)

{

int i=n%10;

int p=1;

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

{

p=p\*j;

}

sum=sum+p;

n=n/10;

}

if(sum==a)

{

System.out.print("YES,it is a Krishnamoorthy number");

}

else

{

System.out.print("NO,it is not a Krishnamoorthy number");

}

}

}

**Program 24: To print the frequency of every character present in the string.**

class FreqCharInaString

{

void quick(String s)

{

s=s.toLowerCase(); //converting the string into lowercase

int l=s.length(); //finding the length of the string

char alph[]=new char[26]; //array for storing alphabets from 'a' to 'z'

int freq[]=new int[26]; //array for storing frequency of all alphabets

char c='a';

for(int i=0; i<26; i++)

{

alph[i]=c; //storing all alphabets from 'a' till 'z' in alph[] array

freq[i]=0; //intializing the count of every alphabet with '0'

c++;

}

char ch;

System.out.println("Output:");

System.out.println("==========================");

System.out.println("Alphabet\tFrequency");

System.out.println("==========================");

/\* Counting frequency of alphabets begins below \*/

for(int i=0; i<26; i++)

{

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

{

ch=s.charAt(j); //extracting characters of the string one by one

if(ch==alph[i]) //first checking the whole string for 'a', then 'b' and so on

freq[i]++; //increasing count of those aplhabets which are present in the string

}

}

for(int i=0; i<26; i++)

{

if(freq[i]!=0) //printing only those alphabets whose count is not '0'

System.out.println(" "+alph[i]+"\t\t "+freq[i]);

}

}

}

**Program 25: A Composite Magic number is a positive integer which is composite as well as a magic number.**

**Composite Number:**

**A composite number is a number that has more than two factors.**

**Magic number:  
A magic number is a number in which the eventual sum of the digits is equal to 1  
For example: 28=2+8=10=1+0=1**

**Accept two positive integers m and n, where m is less than n as user input. Display the number of Composite magic integers that are in the range between m and n (both inclusive) and output them along with the frequency.**

import java.util.\*;

class MagicComposite

{

boolean isComposite(int n) // Function to check for Composite number

{

int count=0;

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

{

if(n%i==0)

count++;

}

if(count>2)

return true;

else

return false;

}

int sumDig(int n) // Function to return sum of digits of a number

{

int s = 0;

while(n>0)

{

s = s + n%10;

n = n/10;

}

return s;

}

boolean isMagic(int n) // Function to check for Magic number

{

int a = sumDig(n);

while(a>9)

{

a = sumDig(a);

}

if(a == 1)

return true;

else

return false;

}

void main()

{

MagicComposite ob = new MagicComposite();

Scanner s=new Scanner(System.in);

System.out.print("Enter the lower limit(m) : ");

int m=s.nextInt();

System.out.print("Enter the upper limit(n) : ");

int n=s.nextInt();

int c=0;

if (m<n)

{

System.out.println("The Composite Magic Integers are: ");

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

{

if(ob.isComposite(i)==true && ob.isMagic(i)==true)

{

if (c==0) // Printing the first number without any comma

System.out.print(i);

else

System.out.print(", "+i);

c++;

}

}

System.out.println(" The frequency of Composite Magic Integers is : "+c);

}

else

System.out.println("OUT OF RANGE");

}

}

**Program 26: An Emirp number is a number which is prime backwards and forwards. Example 13 and 31 are both prime numbers. Thus, 13 is an Emirp number. Design a class Emirp to check if a given number is an Emirp number or not.**

import java.util.\*;

class Emirp

{

int isPrime(int n)

{

for(int i=2;i<n;i++)

{

if(n%i==0)

{

return 0;

}

}

return 1;

}

void isEmirp()

{

Scanner s=new Scanner(System.in);

System.out.println("The No.");

int a=s.nextInt();

int c=a;

int rev=0;

int r=0;

while(c!=0)

{

r=c%10;

rev=rev\*10+r;

c=c/10;

}

int k=isPrime(a);

int x=isPrime(rev);

if(k==1&&x==1)

{

System.out.println("The number"+a+"is Emirp");

}

else

{

System.out.println("Not Emirp");

}

}

}

**Program 27: To check if a date entered is valid or not.**

class ValidDate

{

void main(int month,int day,int year)

{

boolean f=false;

if(month<=12)

{

if(month<=7)

{

if(month%2==0)

{

if(year%4==0)

{

if(month==2)

{

if(day<=29)

{

f=true;

}

else

{

f=false;

}

}

else

{

if(day<=30)

{

f=true;

}

else

{

f=false;

}

}

}

else

{

if(month==2)

{

if(day<=28)

{

f=true;

}

else

{

f=false;

}

}

else

{

if(day<=30)

{

f=true;

}

else

{

f=false;

}

}

}

}

else

{

if(day<=31)

{

f=true;

}

else

{

f=false;

}

}

}

else

{

if(month%2==0)

{

if(day<=31)

{

f=true;

}

else

{

f=false;

}

}

else

{

if(day>30)

{

f=false;

}

else

{

f=true;

}

}

}

}

else

{

f=false;

}

if(f==true)

{

System.out.println("VALID DATE");

}

else

{

System.out.print("INVALID DATE");

}

}

}

**Program 28: The encryption of alphabets are to be done as follows:**

**A=1**

**B=2**

**C=3**

**.**

**.**

**Z=26**

**The potential of a word is found by adding the encrypted value of the alphabets.**

**Example: KITE**

**Potential = 11 + 9 + 20 + 5 = 45**

**Accept a sentence which is terminated by either “ . ” , “ ? ” or “ ! ”. Each word of sentence is separated by single space. Decode the words according to their potential and arrange them in ascending order.**

import java.util.\*;

class WordPotential

{

int findPotential(String s) // Function to find potential of a word

{

s = s.toUpperCase();

int p = 0, l = s.length();

char ch;

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

{

ch = s.charAt(i);

p = p + (ch-64); // if ch = 'A', then 'A'-64 = ASCII value of 'A' - 64 = 65-64 = 1

}

return p;

}

// Function to sort the words in ascending order of their potential

void sortPotential(String w[], int p[])

{

int n = w.length, t1 = 0;

String t2 = "";

for(int i=0; i<n-1; i++)

{

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

{

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

{

t1 = p[i];

p[i] = p[j];

p[j] = t1;

t2 = w[i];

w[i] = w[j];

w[j] = t2;

}

}

}

printResult(w,p);

}

void printResult(String w[], int p[]) // Function to print the final result

{

int n = w.length;

String ans = "";

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

{

ans = ans + " " + w[i];

}

System.out.println("\nOutput\t\t : \t"+ans);

}

void main()

{

WordPotential ob = new WordPotential();

Scanner sc = new Scanner(System.in);

System.out.print("Enter a sentence : \t");

String s = sc.nextLine();

StringTokenizer str = new StringTokenizer(s," .,?!");

int n = str.countTokens();

String words[] = new String[n];

int potential[] = new int[n];

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

{

words[i] = str.nextToken(); // Saving words one by one in an array

potential[i] = ob.findPotential(words[i]); // Saving potential of every word

}

// Printing the words along with their potential

System.out.print("\nPotential\t : \t");

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

{

System.out.println(words[i]+"\t= "+potential[i]);

System.out.print("\t\t\t");

}

ob.sortPotential(words,potential);

}

}

**Program 29: Write a program to declare a square matrix A[ ][ ] of order MxM where ‘M’ is the number of rows and the number of columns, such that M must be greater than 2 and less than 10. Accept the value of M as user input. Display an appropriate message for an invalid input. Allow the user to input integers into this matrix.**

**Perform the following tasks:**

**a) Display the original matrix.**

**b) Rotate the matrix 90° clockwise**

**c) Find the sum of the elements of the four corners of the matrix.**

import java.util.\*;

class RotationOfMatrix

{

void main()

{

Scanner sc=new Scanner(System.in);

System.out.print("Enter the size of the matrix : ");

int m=sc.nextInt();

if(m<3 || m>9)

System.out.println("Size Out Of Range");

else

{

int A[][]=new int[m][m];

/\* Inputting the matrix \*/

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

{

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

{

System.out.print("Enter an element : ");

A[i][j]=sc.nextInt();

}

}

/\* Printing the original matrix \*/

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

System.out.println("The Original Matrix is : ");

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

{

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

{

System.out.print(A[i][j]+"\t");

}

System.out.println();

}

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

/\*Rotation of matrix begins here \*/

System.out.println("Matrix After Rotation is : ");

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

{

for(int j=m-1;j>=0;j--)

{

System.out.print(A[j][i]+"\t");

}

System.out.println();

}

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

int sum = A[0][0]+A[0][m-1]+A[m-1][0]+A[m-1][m-1]; // Finding sum of corner elements

System.out.println("Sum of the corner elements = "+sum);

}

}

}

**Program 30: Write a program to declare a square matrix A[ ][ ] of order ‘n’. Allow the user to input positive integers into this matrix. Perform the following tasks on the matrix:**

**(i) Output the original Matrix**

**(ii) Find the SADDLE POINT for the matrix. If the matrix has no saddle point, output the message “NO SADDLE POINT”.**

**[Note: A saddle point is an element of the matrix such that it is the minimum element for the row to which it belongs and the maximum element for the column to which it belongs. Saddle point for a given matrix is always unique.]**

import java.util.\*;;

class SaddlePoint

{

void main()

{

Scanner s=new Scanner(System.in);

System.out.print("Enter the order of the matrix : ");

int n=s.nextInt();

int A[][]=new int[n][n];

System.out.println("Inputting the elements in the matrix");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"); // Ignore these. They are just for styling

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

{

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

{

System.out.print("Enter Element at ["+i+"]["+j+"] : ");

A[i][j]=s.nextInt();

}

}

/\* Printing the Original Matrix \*/

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

System.out.println("The Original Matrix is");

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

{

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

{

System.out.print(A[i][j]+"\t");

}

System.out.println();

}

int max, min;

int temp=0;

int saddle=0;

boolean flag=false;

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

{

/\* Finding the minimum element of a row \*/

min = A[i][0]; // Initializing min with first element of every row

temp = 0;

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

{

if(A[i][j]<min)

{

min = A[i][j];

temp = j; // Saving the column position of the minimum element of the row

}

}

/\* Finding the maximum element in the column

\* corresponding to the minimum element of row \*/

max = A[0][temp]; // Initializing max with first element of that column

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

{

if(A[k][temp]>max)

{

max = A[k][temp];

}

}

/\* If the minimum of a row is same as maximum of the corresponding column,

then, we have that element as the Saddle point \*/

if(max==min)

{

flag=true;

saddle=max;

break;

}

}

if(flag==true)

{

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

System.out.println("Saddle Point:"+saddle);

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

}

else

System.out.println("No Saddle Point");

}

}