PROGRAM 12:

A PROGRAM TO PERFORM BINARY SEARCH.

aLGORITHM:

Step 1: Start.

Step 2: Take an array as input.

Step 3: Enter the element to be searched.

Step 4: Take three variables for first, last and middle index of the array.

Step 5: Check the middle element of the array and the element to be searched.

* If the elements are equal, print “Search Successful”.
* If the middle element is greater, decrease the middle index value by 1 and check again for equality.
* If the middle element is smaller, increase the middle index value by 1 and check again for equality.

Step 6: If the first index becomes greater than the last index, print “Search Unsuccessful”.

Step 7: End.

Program code:

import java.util.\*;

class Binary\_Recur

{

Scanner sc=new Scanner(System.in);

int a[]=new int[10];

int Search(int u, int l, int ele)

{

int mid=(u+l)/2;

if(u>l)

return -1;

else if(ele==a[mid])

return mid;

else if(ele>a[mid])

return Search(mid+1,l,ele);

else

return Search(u,mid-1,ele);

}

void display()

{

System.out.println("Enter the array elements");

for(int i=0;i<10;i++)

a[i]=sc.nextInt();

System.out.println("Enter the element to be searched");

int ele=sc.nextInt();

int f=Search(0,9,ele);

if(f!=(-1))

System.out.println("Search Successful, Present at position of "+(f+1));

else

System.out.println("Search Unsuccessful");

}

void main()

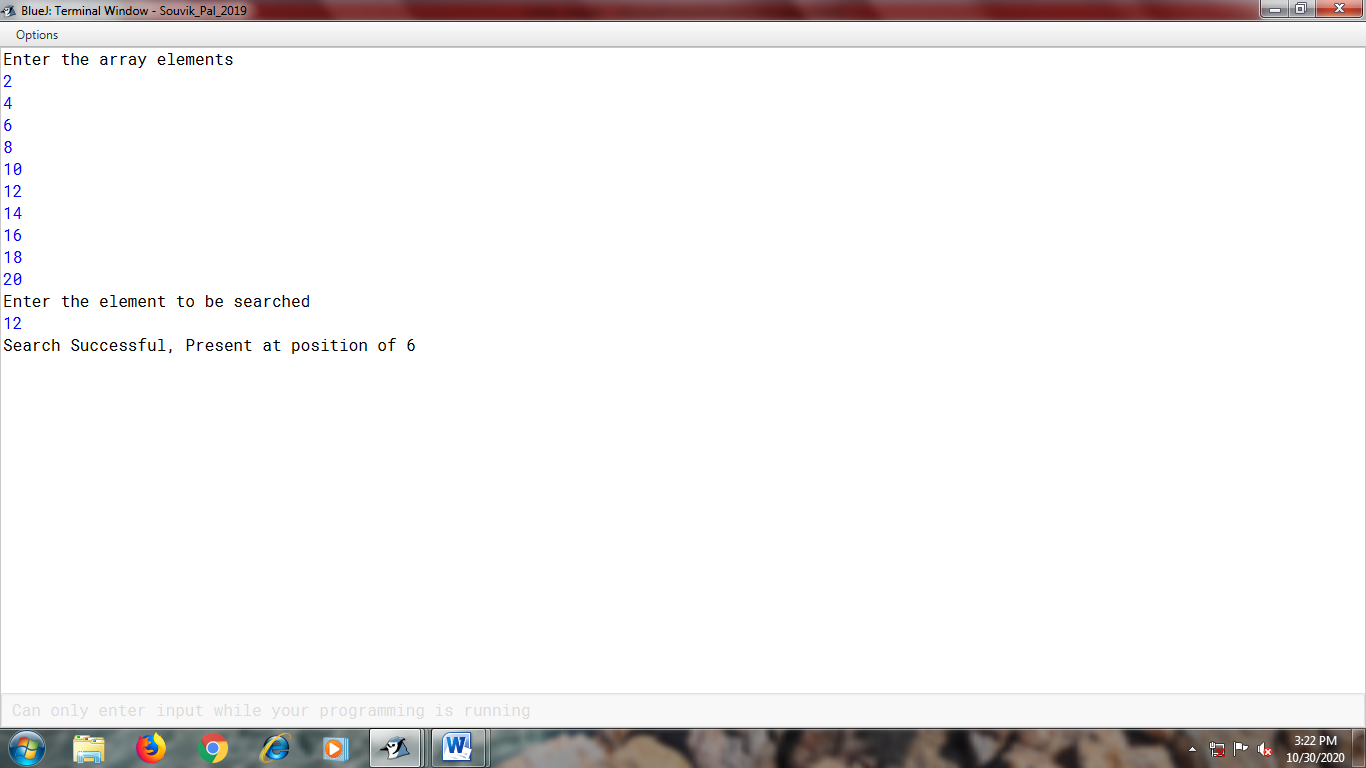
{

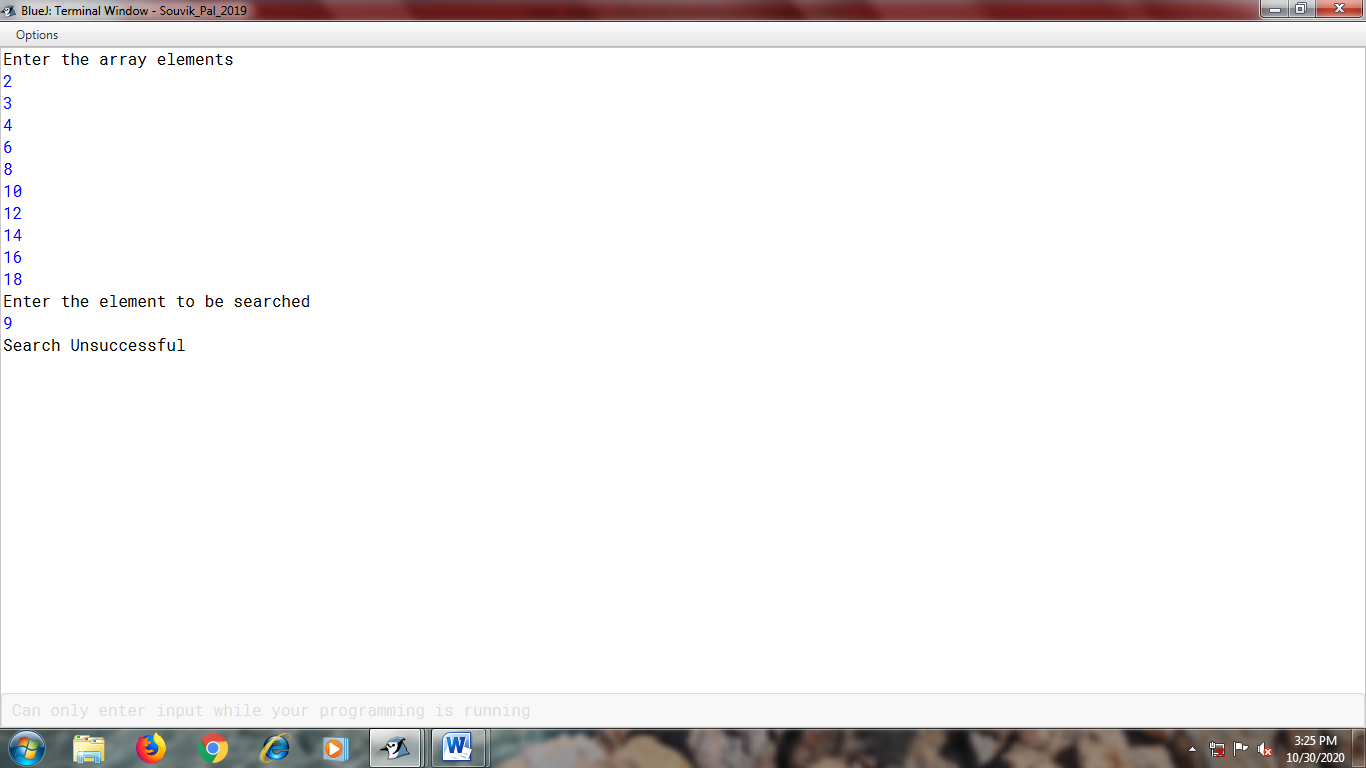
display();

}

}

Output:





PROGRAM 13:

A PROGRAM TO COUNT AND PRINT DOUBLE LETTER SEQUENCES PRESENT IN A SENTENCE.

aLGORITHM:

Step 1: Start.

Step 2: Take a sentence as input.

Step 3: Using recursion, process a series through the sentence, till the second last index of the sentence.

Step 4: Pick up the characters one by one, followed by its succeeding character.

Step 5: If they are equal, increase the counter variable by 1.

Step 6: At the end of the series, print the value of the counter variable.

Step 7: End.

PROGRAM CODE:

import java.util.\*;

class Double\_Letter

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

int c;

void main()

{ System.out.println("Enter a sentence");

String s=sc.nextLine( ).toUpperCase( );

letter(s,0);

System.out.println("Count="+c);

}

void letter(String s,int i)

{ if(i<s.length()-1)

{

char ch1=s.charAt(i);

char ch2=s.charAt(i+1);

if(ch1==ch2)

c++;

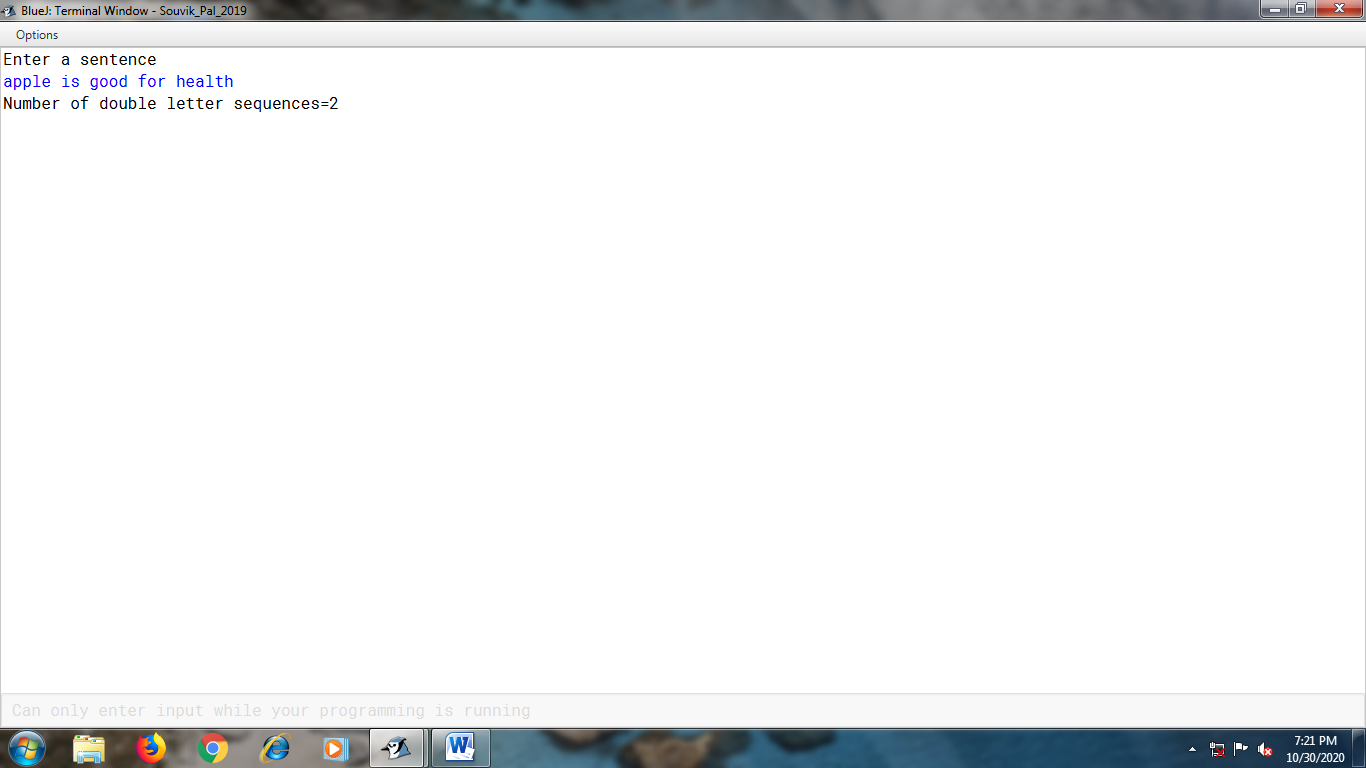
letter(s,i+1);

}

}

}

OUTPut:



Program 14:

A program to find perimeter and area of a parallelogram (rectangle).

ALgorithm:

Step 1: Start.

Step 2: Create a **super** class ‘Perimeter’.

Step 3: With the help of a constructor, calculate and print the perimeter of the parallelogram (rectangle).

Step 4: Call the calculate() function and print the length, breadth and perimeter in the show() function.

Step 5: Create a **sub** class ‘Area’.

Step 6: With the help of a constructor and super(), bring the values from the super class Perimeter and calculate the area in the cal\_area() function.

Step 7: Display the perimeter and area of the rectangle using super() .

Step 8: In the main() function, take the length and breadth as input, create an object and call the required functions.

Step 9: End.

PROGRAM CODE:

Perimeter:-

import java.util.\*;

class Perimeter

{

int a;

int b;

Perimeter(int x,int y)

{

a=x;

b=y;

}

int calculate()

{

int P=2\*(a+b);

return P;

}

void show()

{

int p=calculate();

System.out.println("Length="+a);

System.out.println("Breadth="+b);

System.out.println("Perimeter of a rectangle="+p);

}

}

Area:-

import java.util.\*;

class Area extends Perimeter

{

int a,b;

double area;

Area(int x,int y)

{

super(x,y);

a=x;

b=y;

}

public void cal\_area()

{

area=a\*b;

}

void show()

{

super.show();

System.out.println("Area="+area);

}

static void main()

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the length and breadth");

int l=sc.nextInt();

int b=sc.nextInt();

Area obj=new Area(l,b);

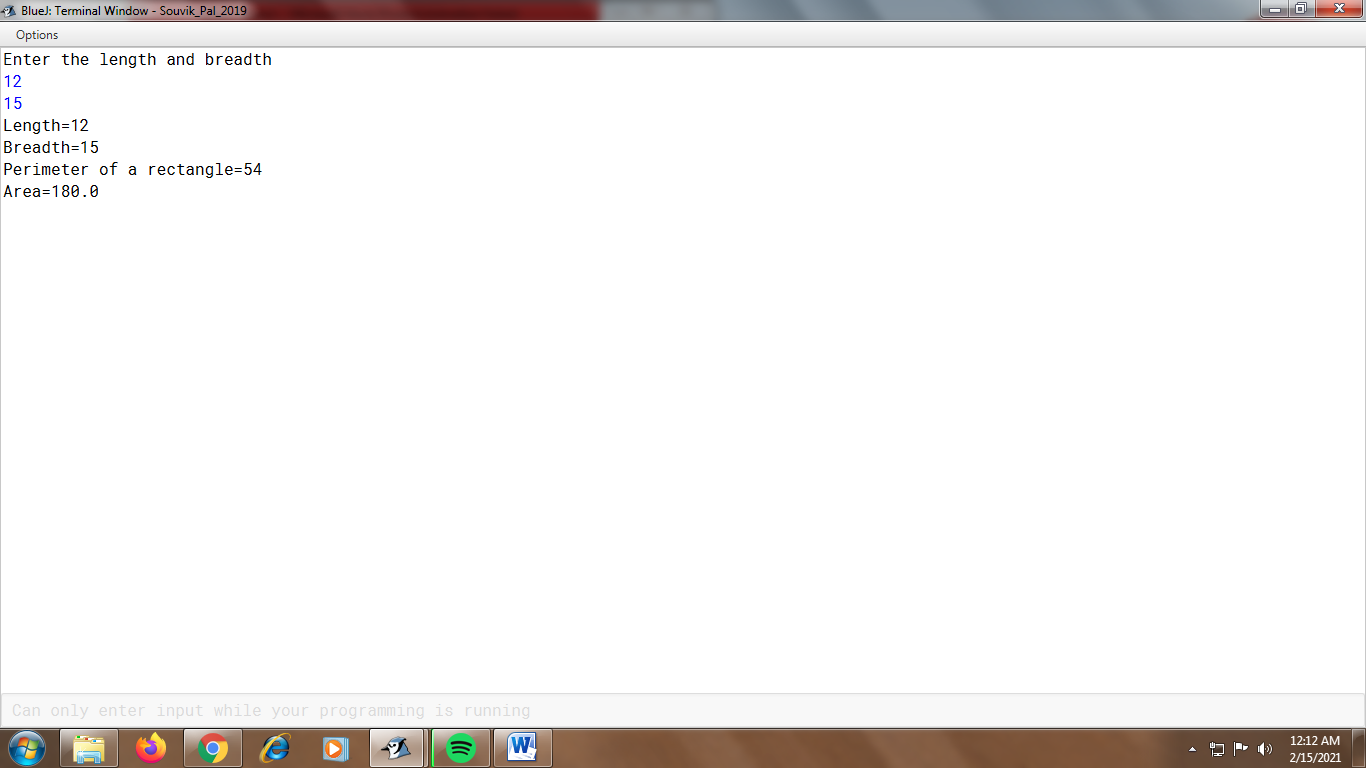
obj.cal\_area();

obj.show();

}

}

OUTPUT:



Program 15:

A Program to calculate the Half-yearly Sale and Annual Sale.

ALGORITHM:

Step 1: Start.

Step 2: Create a **super** class ‘Product’.

Step 3: Using a constructor, take the Product name and code as input in the read\_info() function.

Step 3: Create a **sub** class ‘HalfyearlySale’.

Step 4: Take the half-yearly sale amount as input in getSale() function.

Step 5: Calcualte the profit in the getProfit() function.

Step 6: In the display() function, print the Product name, code, Half-Yearly sale and profit.

Step 7: Create another **sub** class ‘AnnualSale’.

Step 8: Take the annual sale amount as input in getSale() function.

Step 9: Calcualte the profit in the getProfit() function.

Step 10: In the display() function, print the Product name, code, Annual sale and profit.

Step 11: In the main() function, create 2 objects, one of ‘HalfyearlySale’, the other of ‘AnnualSale’, and call the required functions.

Step 12: End.

PROGRAM COde:

Product:-

import java.util.\*;

class Product

{

Scanner sc=new Scanner(system.in);

Scanner sc1=new Scanner(system.in);

protected String product\_name;

protected long code;

Product()

{

product\_name="";

code=0;

}

void read\_info()

{

System.out.println("Enter Product name and code:");

product\_name=sc.nextLine();

code=sc1.nextLong();

} }

HalfyearlySale:-

import java.util.\*;

class Halfyearlysale extends Product

{

double sale,profit;

void getSale()

{

System.out.println("Enter the half-yearly sale amount");

sale=sc.nextDouble();

}

double getProfit()

{

profit=(20.0/100.0)\*sale;

return profit;

}

void display()

{

profit=getProfit();

System.out.println("Product Name is: "+product\_name);

System.out.println("Code is: "+code);

System.out.println("Half-Yearly sale="+sale);

System.out.println("Profit="+profit);

}

}

AnnualSale:-

import java.util.\*;

class Annualsale extends Product

{

static double sale, profit;

void getSale()

{

System.out.println("Enter annual sale amount");

sale=sc.nextDouble();

}

double getProfit()

{

profit=(40.0/100.0)\*sale;

return profit;

}

void display()

{

profit=getProfit();

System.out.println("Product name: "+product\_name);

System.out.println("Code: "+code);

System.out.println("Annual sale: "+sale);

System.out.println("Profit: "+profit);

}

static void main()

{

HalfyearlySale ob1=new HalfyearlySale();

ob1.read\_info();

ob1.getSale();

ob1.display();

AnnualSale ob2=new AnnualSale();

ob2.product\_name=ob1.product\_name;

ob2.code=ob1.code;

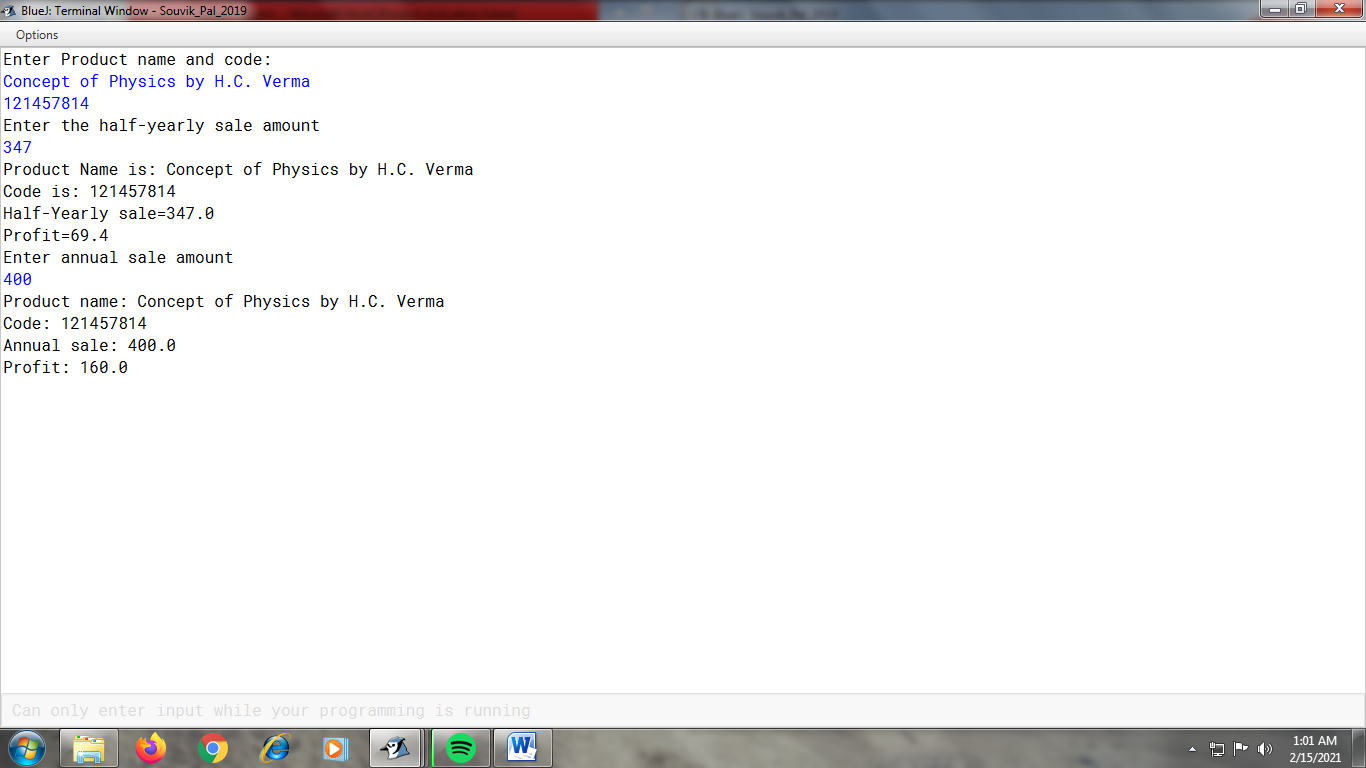
ob2.getsale();

ob2.display();

}

}

Output:



Program 16:

A Program to display A Student’s details, marks and final result. .

ALGOrithm:

Step 1: Start.

Step 2: Create a **super** class ‘Student’.

Step 3: Using a constructor, in the accept() function,

Take the student’s name and roll number as input.

Step 4: Print the name of the student and the roll number in the show() function.

Step 5: Create a **sub** class ‘Marks’.

Step 6: Using the constructor, input the marks in the getMarks() function.

Step 7: Print the two marks in the dispMarks() function.

Step 8: Create another **sub** class ‘Result’.

Step 9: In the display() function, calculate the total marks and display the Student’s name, roll number, marks and the total mark.

Step 10: In the main() function, create 3 objects, one of ‘Student’, one of ‘Marks’, one of “Result” and call the required functions.

Step 11: End.

Program code:

Student:-

import java.util.\*;

class Student

{

String std;

int roll;

Student()

{

std="";

roll=0;

}

void accept()

{

Scanner sc=new Scanner(System.in);

Scanner sc1=new Scanner(System.in);

System.out.println("Enter the student's name and roll number:");

std=sc1.nextLine();

roll=sc.nextInt();

}

void show()

{

System.out.println("Student's name: "+std);

System.out.println("Roll number: "+roll);

}

}

Marks:-

import java.util.\*;

class Marks extends Student

{

protected float a;

protected float b;

Marks()

{

a=0.0F;

b=0.0F;

}

void getMarks(float x, float y)

{

a=x;

b=y;

}

void dispMarks()

{

System.out.println("Marks in the 2 semesters are as follows:");

System.out.println(a);

System.out.println(b);

}

}

Result:-

import java.util.\*;

class Result extends Marks

{

float total;

void display()

{

total=a+b;

System.out.println("Student's name: "+std);

System.out.println("Roll number: "+roll);

System.out.println("Marks in the 2 semesters are as follows:");

System.out.println(a);

System.out.println(b);

System.out.println("Total marks of computer in the semester: "+total);

}

void main()

{

Student ob1=new Student();

Result ob3=new Result();

ob3.accept();

Scanner sc=new Scanner(System.in);

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

float m=sc.nextFloat();

float n=sc.nextFloat();

ob3.getMarks(m,n);

ob3.display();

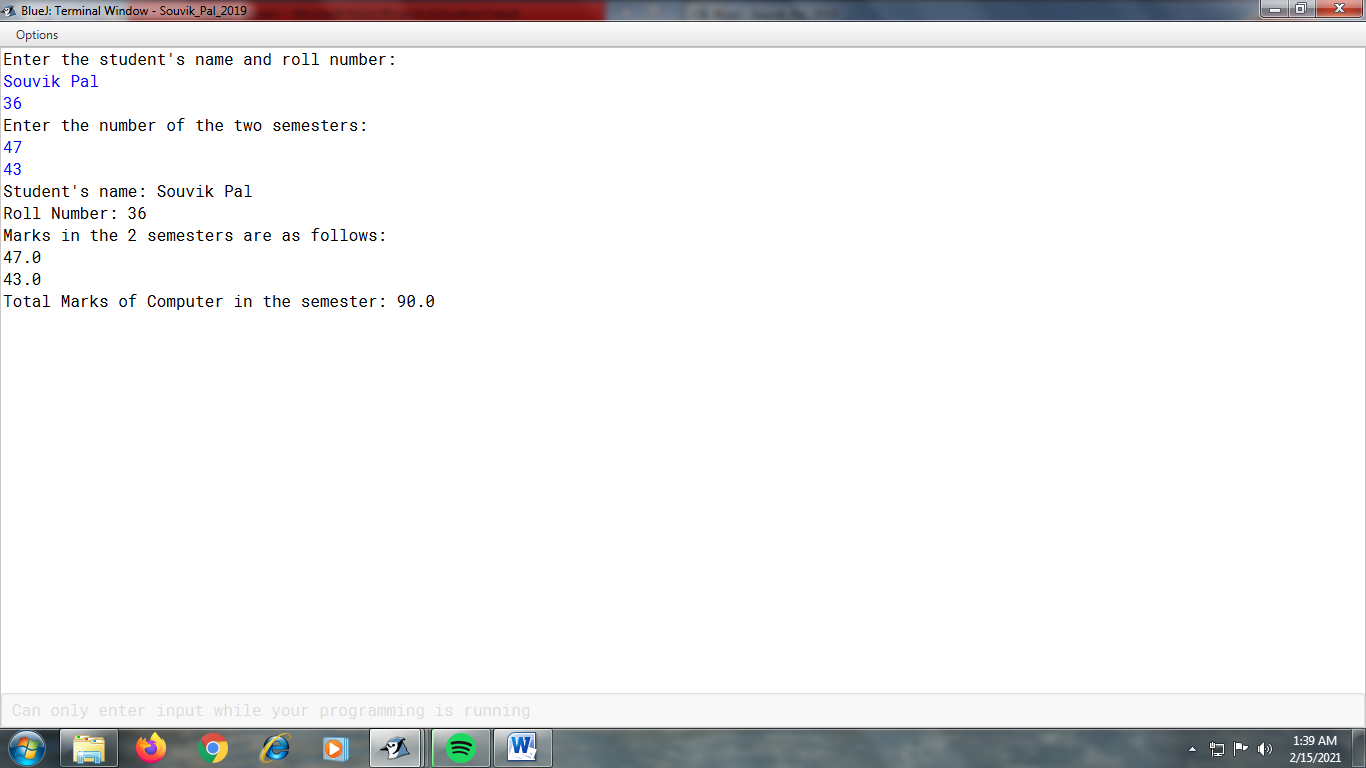
Marks ob2=new Marks();

ob2.getMarks(m,n);

}

}

OUtput:



Program 17:

A Program to find the square and cube of a number.

Algorithm:

Step 1: Start.

Step 2: Create an **abstract** class ‘Num’, and create an **abstract** calculate() function within it.

Step 3: Create a **sub** class ‘Square’.

Step 4: Within it , in the the calculate() function, calculate and print the square of the number.

Step 5: Create another **sub** class ‘Cube’.

Step 6: Within it , in the the calculate() function, calculate and print the cube of the number.

Step 7: Create a class ‘output’.

Step 8: In the main() function, take the number as input , create two objects, one of ‘Square’ and the other of ‘Cube’ and call the required functions.

Step 9: End.

Program code:

Num:-

import java.util.\*;

public abstract class Num

{

abstract void calculate(double x);

}

class Square extends Num

{

void calculate(double x)

{

double s=x\*x;

System.out.println("The square is="+s);

}

}

class cube extends num

{

void calculate(double x)

{

double s=x\*x\*x;

System.out.println("The cube is="+s);

} }

Output:-

import java.util.\*;

class Output

{

void main()

{

Scanner sc=new Scanner(System.in);

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

double n=sc.nextDouble();

Square ob=new Square();

ob.calculate(n);

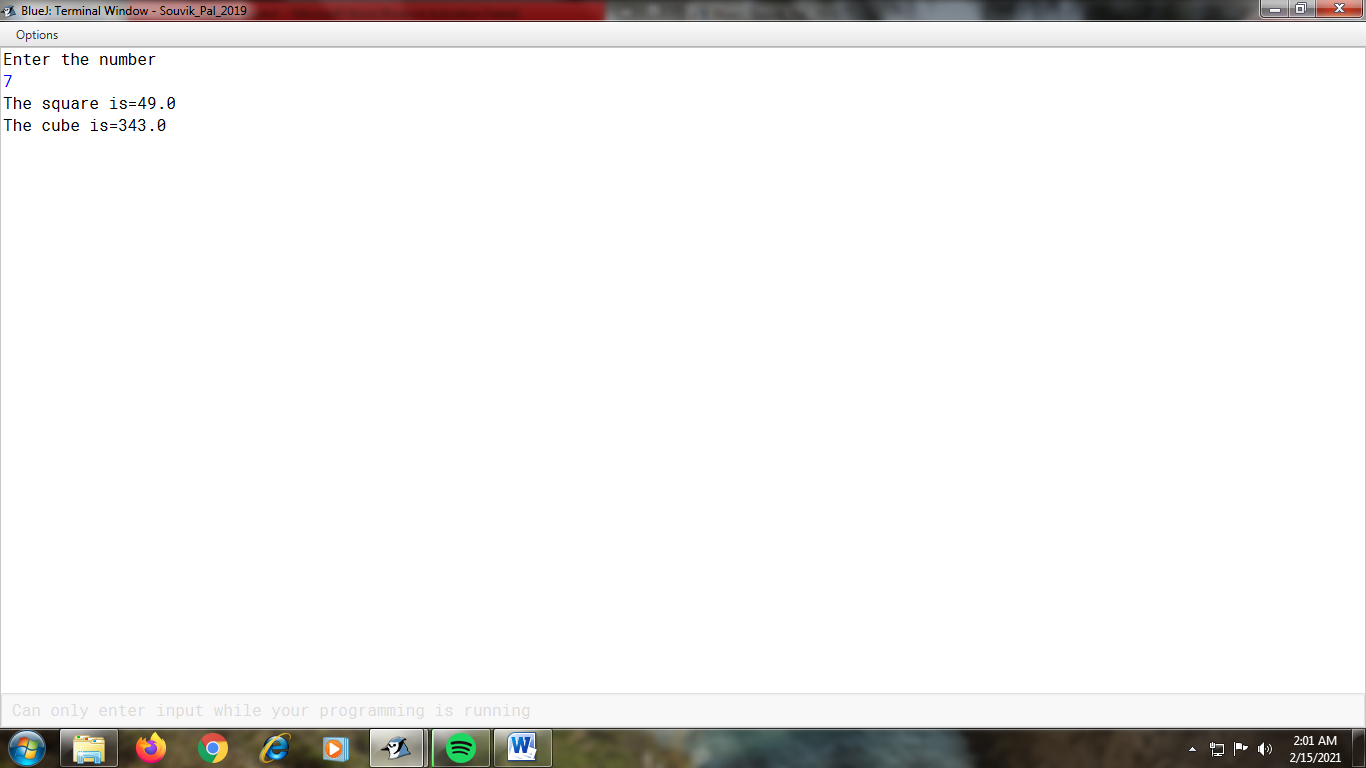
Cube ob1=new Cube();

ob1.calculate(n);

}

}

Output:



Program 18:

A Program to implement stack using array.

ALgorithm:

Step 1: Start.

Step 2: Enter the capacity of the array and initialize the ‘top’ variable with ‘-1’.

Step 3: Using a ‘while’ loop and ‘switch’ case, create the options for the user to choose.

Step 4: In the insert( ) function, check whether the value of the ‘top’ variable is equal to one less than the array’s length. If so, print “Stack is Full”, else insert the element in the array, and print the inserted element.

Step 5: In the delete( ) function, check whether the ‘top’ variable is less than 0. If so, print “Stack Underflow” and return 0, else print the deleted element from the array, and return the deleted element.

Step 6: In the display( ) function, check whether the ‘top’ variable is less than 0. If so, print “Stack Underflow”, else print the elements present in the array (stack).

Step 6: End.

PROGram code:

import java.util.\*;

class Stack

{

int top,arr[];

Stack(int size)

{

top=-1;

arr=new int[size];

}

void insert(int data)

{

if(top==arr.length-1)

System.out.println("Stack is Full");

else

{

arr[++top]=data;

System.out.println("Pushed data="+arr[top]);

} }

int delete()

{

if(top<0)

{

System.out.println("Stack Underflow");

return 0;

}

else

{

System.out.println("Popped Data="+arr[top]);

return arr[top--];

} }

void display()

{

if(top<0)

System.out.println("Stack Underflow");

else

{

System.out.println("Stack Elements are:");

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

System.out.println(arr[i]);

} }

void main()

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

System.out.println("Enter the capacity of the array:");

int s=sc.nextInt();

int ch=0;

Stack stk=new Stack(s);

while(ch!=4)

{

System.out.println("1. Insert Elements. \n2. Delete Elements. \n3.

Display Elements. \n4. Exit. \n\nEnter your choice.");

ch=sc.nextInt();

switch(ch)

{

case 1:System.out.println("Enter data:");

int ele=sc.nextInt();

stk.insert(ele);

break;

case 2:stk.delete();

break;

case 3:stk.display();

break;

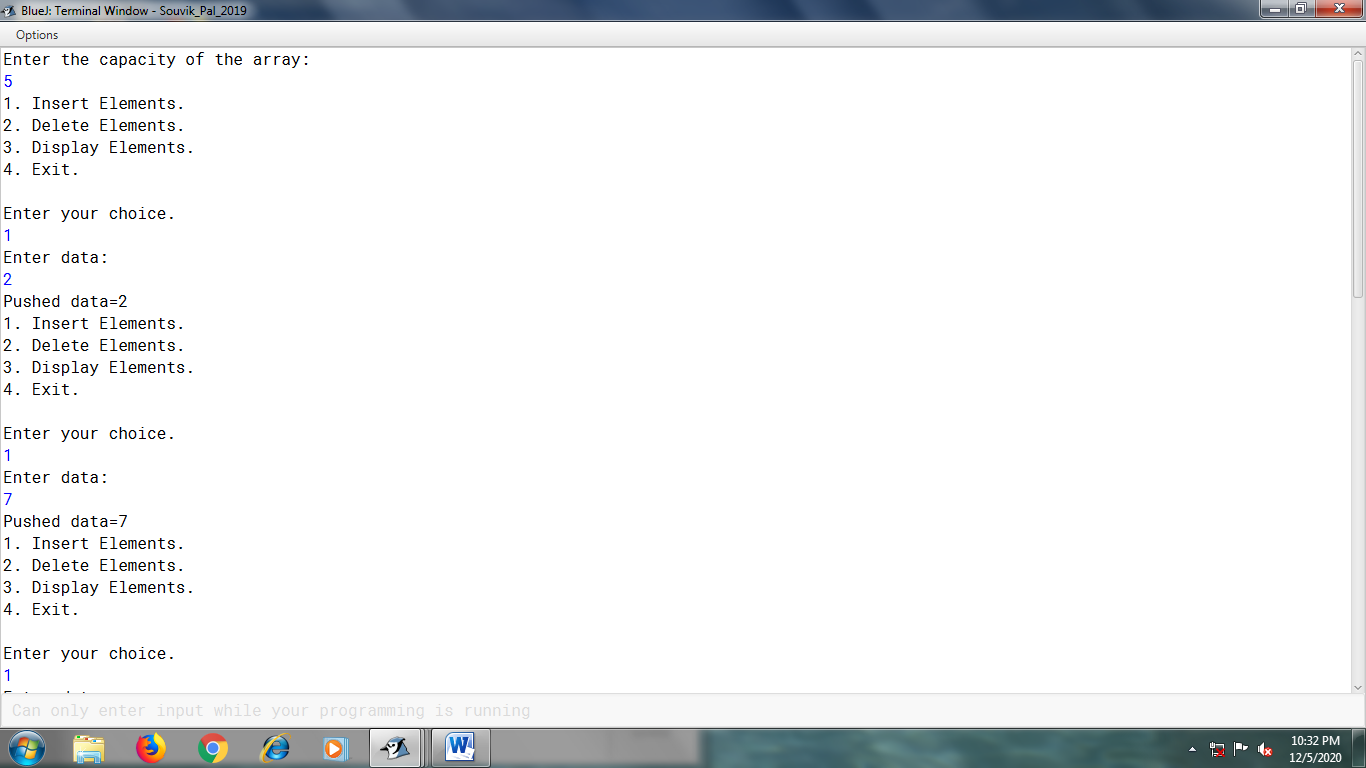
case 4:break;

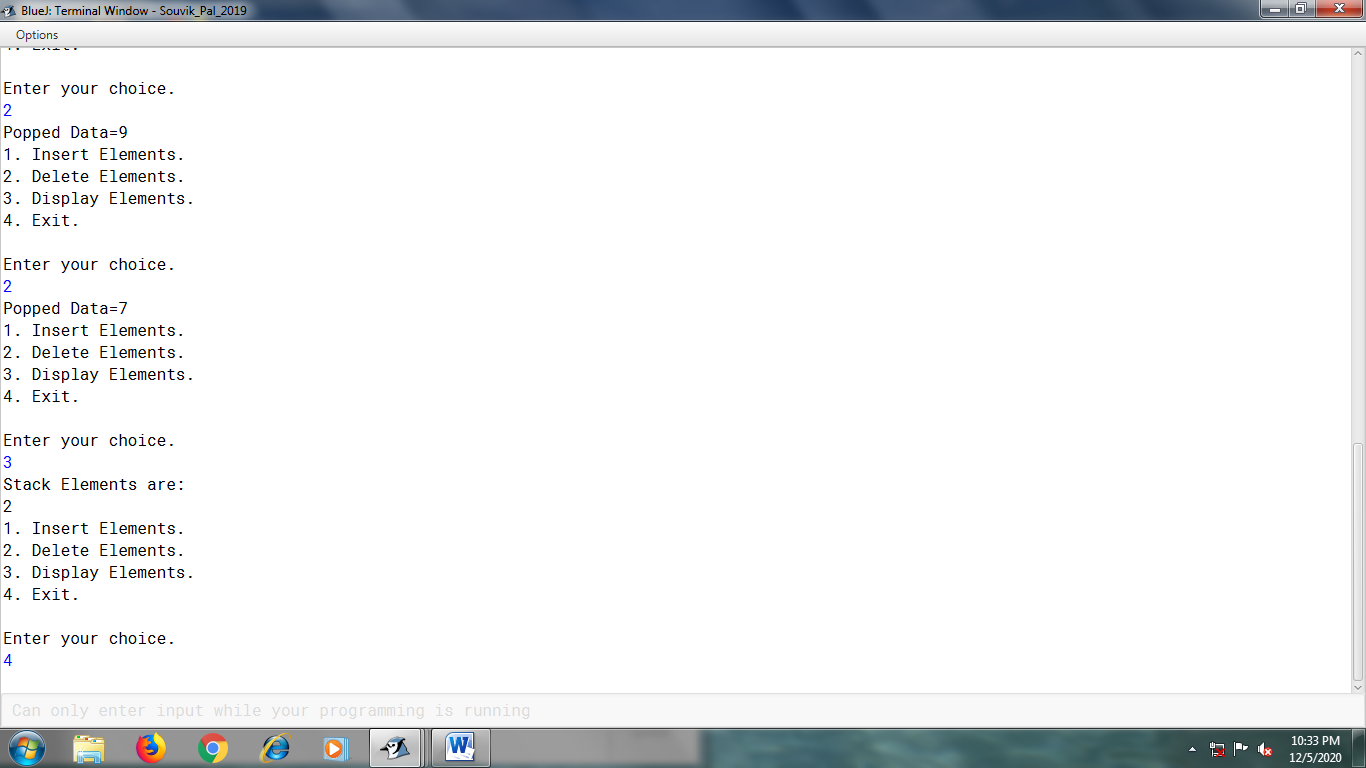
default:System.out.println("Invalid Input");

} }

} }

Output :





PROgram 19:

A program to implement queue using array.

Algorithm:

Step 1: Start.

Step 2: Enter the capacity of the array and initialize the ‘front’ and ‘rear’ variable with ‘-1’.

Step 3: Using a ‘while’ loop and ‘switch’ case, create the options for the user to choose.

Step 4: In the insert( ) function, check whether the difference between ‘rear’ and ‘front’ is equal to one less than the size of the array, or check whether ‘front’ is equal to 0 and ‘rear’ is equal to one less than the size of the array. If so, print “Overflow”, else check whether the ‘front’ and ‘rear’ value is equal to -1. If so, increment the rear and front value and enter the new element into the array (queue), else increment only the ‘rear’ value by 1 and enter the new element into the array (queue).

Step 5: In the delete( ) function, check whether the ‘front’ and ‘rear’ values are both equal to ‘-1’. If so, print “Empty”, else check whether the front value is greater than the rear value. If so, print “Empty”, else print the deleted element and increment the ‘front’ value by 1.

Step 6: In the display( ) function, check whether the ‘front’ and ‘rear’ values are equal to ‘-1’ or ‘front’ value is greater than the size of the array (queue). If so, print “Empty”, else print the elements present in the array (queue).

Step 7: End.

Program code:

import java.util.\*;

class qdata

{

int rear=-1;

int front=-1;

int size;

int queue[];

void main()

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the size of the array");

size=sc.nextInt();

queue=new int[size];

boolean f=true;

while(f)

{

System.out.println("\n1. Insert Elements.");

System.out.println("2. Delete Elements.");

System.out.println("3. Display Elements.");

System.out.println("4. Exit.");

System.out.println("Enter your choice:");

int ch=sc.nextInt();

switch(ch)

{

case 1:

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

int num=sc.nextInt();

insert(num);

break;

case 2:

delete();

break;

case 3:

display();

break;

default:

f=false;

break;

}

}

}

void insert(int n)

{

if((rear-front==(size-1)) || (front==0 && rear==size-1))

System.out.println("Overflow");

else if(front ==-1 && rear==-1)

{

rear++;

front++;

queue[rear]=n;

}

else

{

rear++;

queue[rear]=n;

}

}

void delete()

{

if(front==-1 && rear==-1)

System.out.println("Empty");

else if(front>rear)

System.out.println("Empty");

else

{

System.out.println("Deleted element="+queue[front]);

front++;

}

}

void display()

{

if((rear==-1 && front==-1) || front>size)

System.out.println("Empty");

else

System.out.println("Elements present are:");

for(int i=front;i<=rear;i++)

{

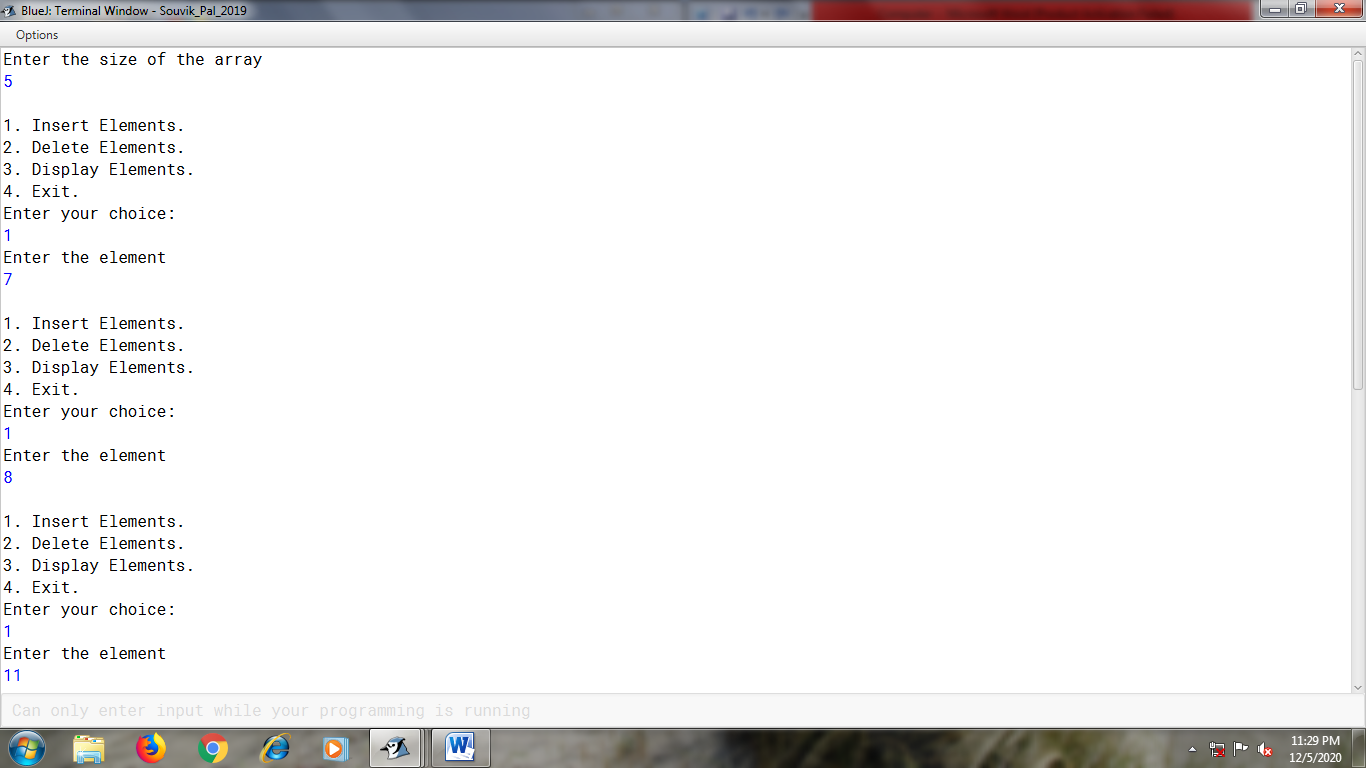
System.out.println(" "+queue[i]);

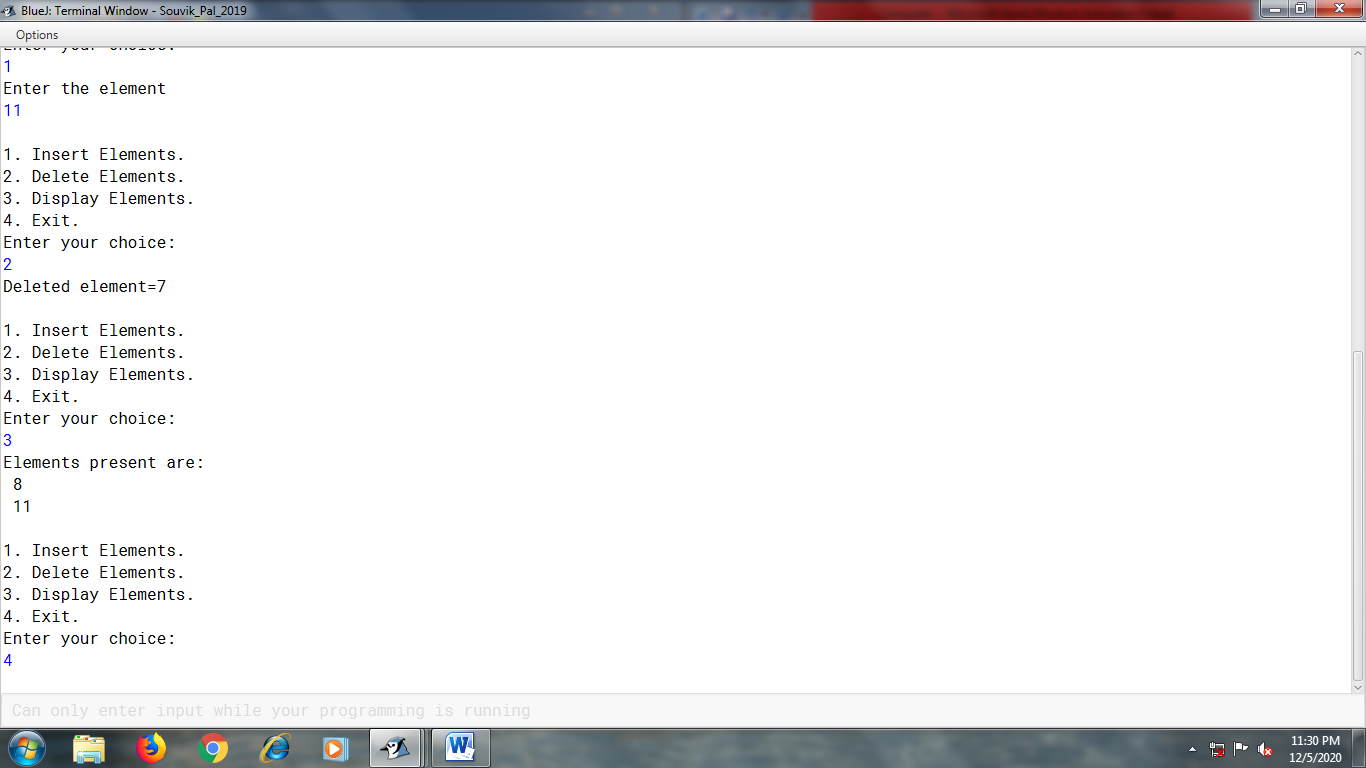
}

}

}

Output:





Program 20:

A program to find number of words in a sentence which are beginning and ending with vowels.

Algorithm:

Step 1: Start.

Step 2: Take the sentence as input.

Step 3: Convert the sentence to uppercase and initialize the counter variable with 0.

Step 4: Using String Tokenizer, pick up the words of the sentence and from the words, pick up the first and last character.

Step 6: Check whether the first and last characters are vowels or not. If so, increment the counter variable by 1.

Step 7: After the while loop terminates, print “Number of words beginning and ending with vowels” and the counter variable.

Step 8: End

Program code:

import java.util.\*;

class Count

{

void main()

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter the sentence:");

String s=sc.nextLine().toUpperCase();

StringTokenizer st=new StringTokenizer(s);

int c=0;

while(st.hasMoreTokens())

{

String w=st.nextToken();

char ch=w.charAt(0);

char ch1=w.charAt(w.length()-1);

if((ch=='A' || ch=='E' || ch=='I' || ch=='O' |ch=='U')&& (ch1=='A' || ch1=='E' || ch1=='I' || ch1=='O' || ch1=='U'))

c++;

}

System.out.println("Number of words beginning and ending with vowels: "+c);

} }

Output:

