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()

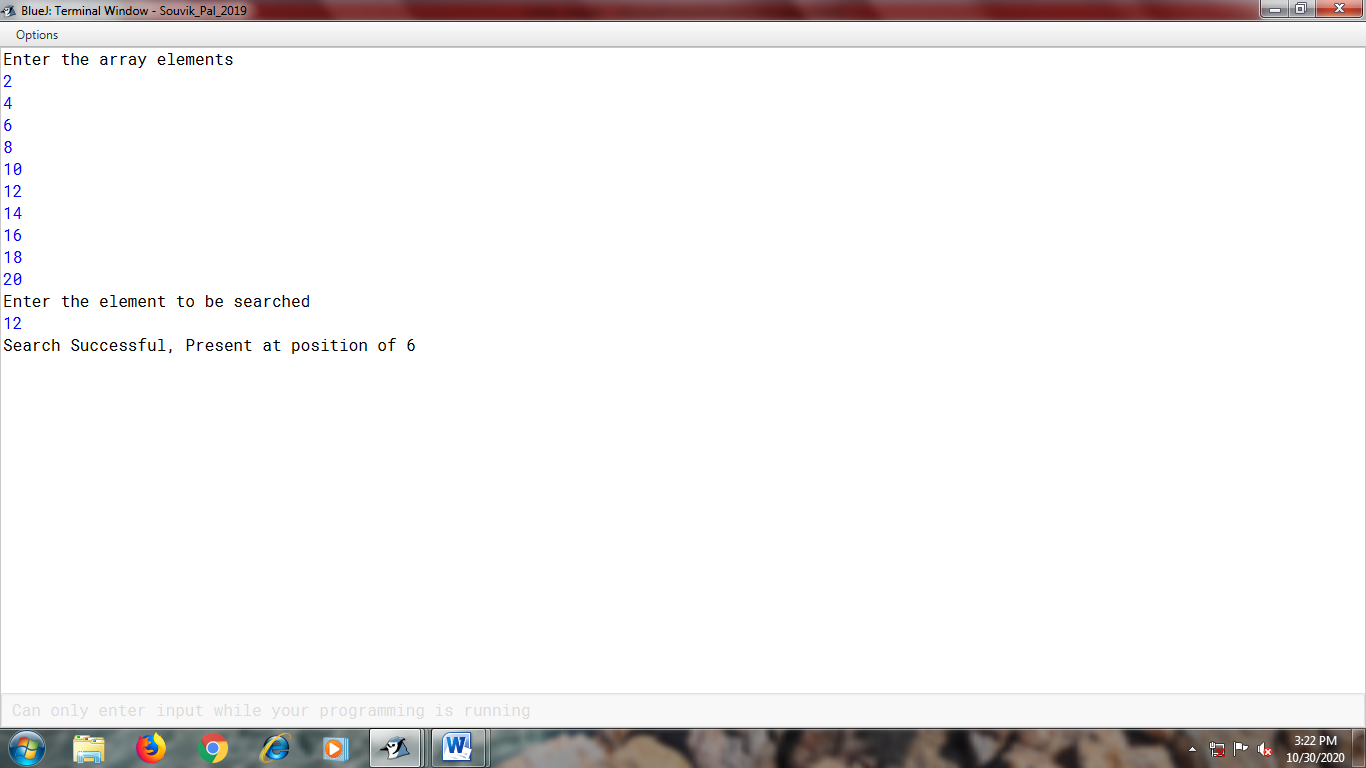
{

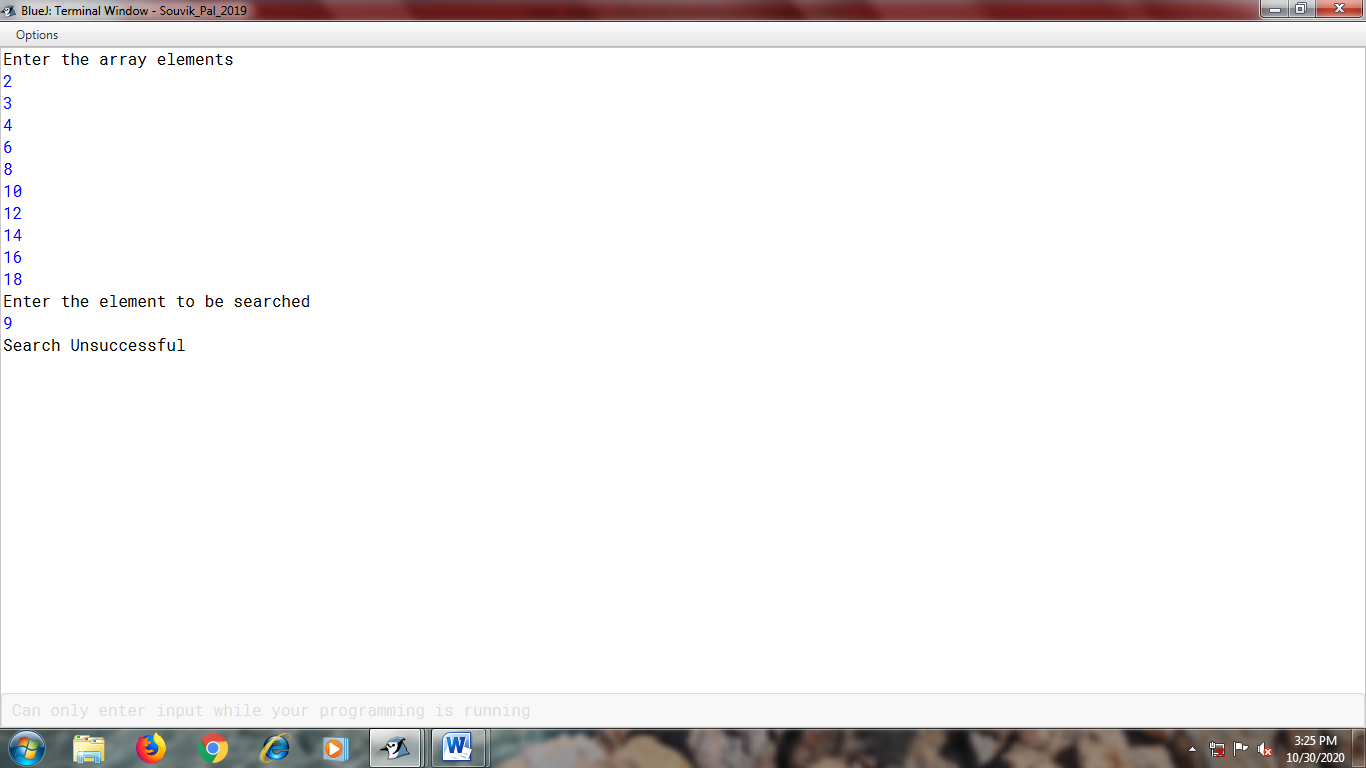
display();

}

}

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++;

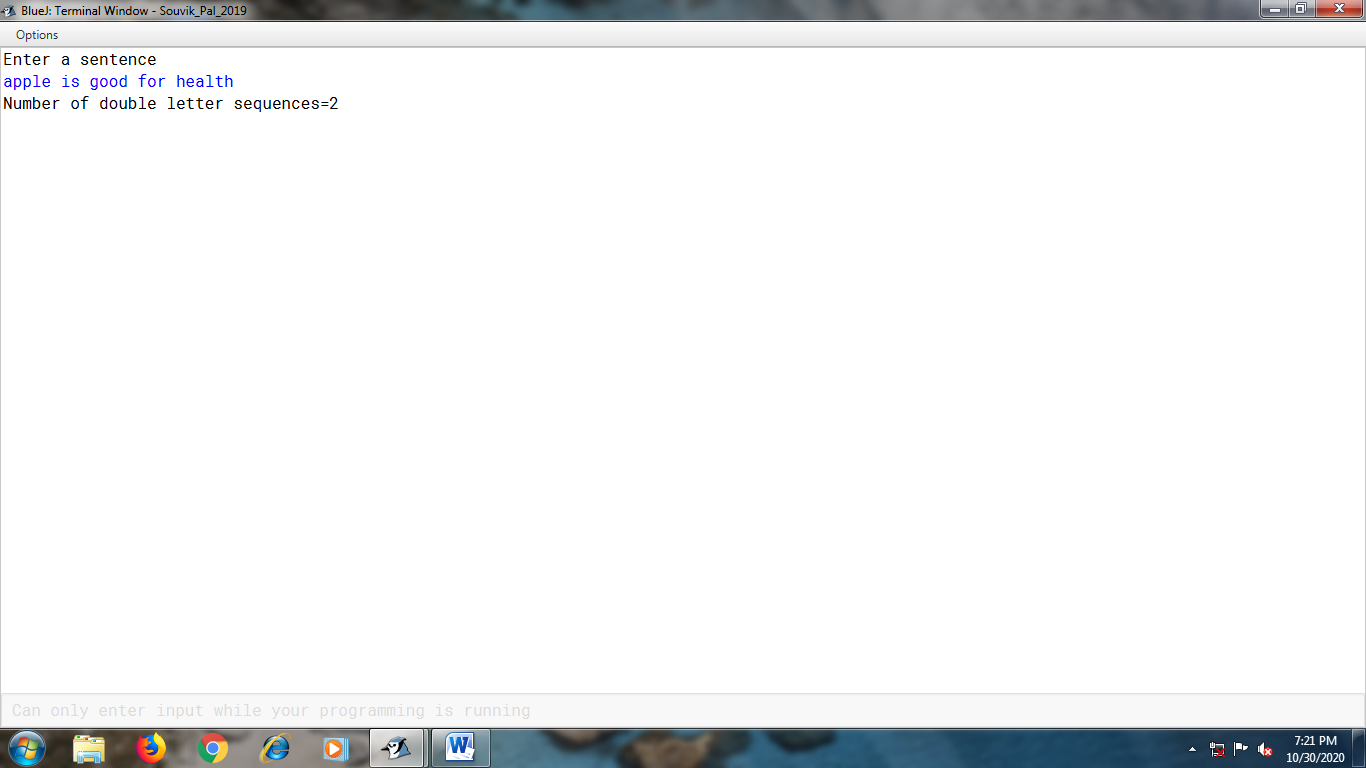
letter(s,i+1);

}

}

}

OUTPut:



Program 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;

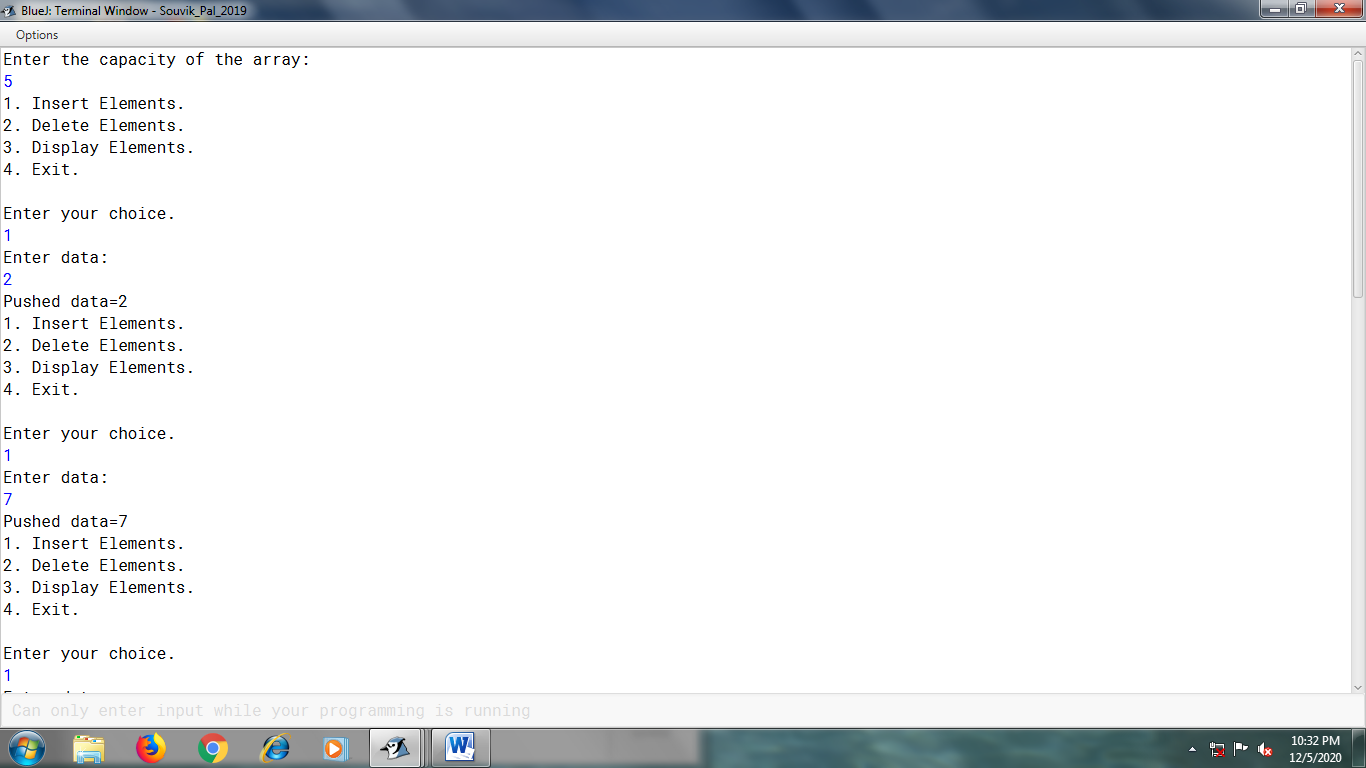
case 4:break;

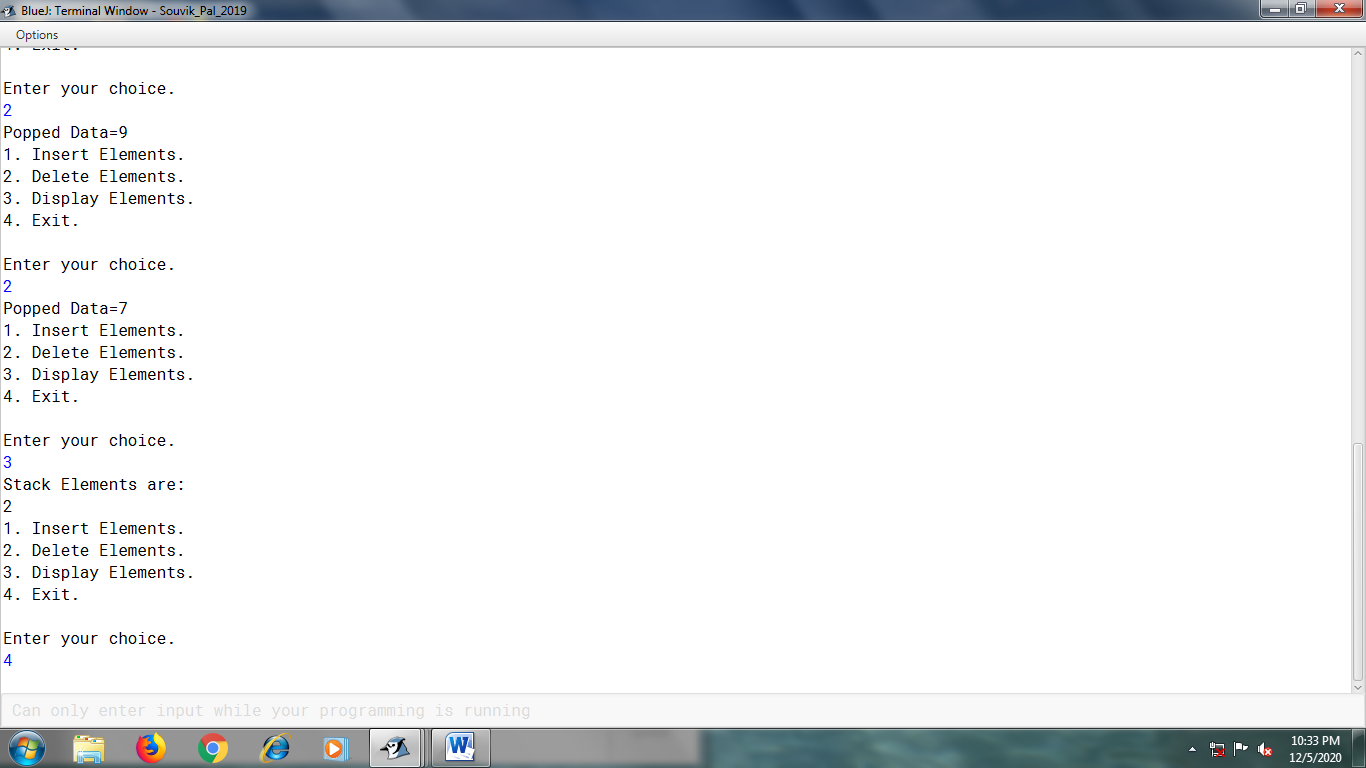
default:System.out.println("Invalid Input");

} }

} }

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++)

{

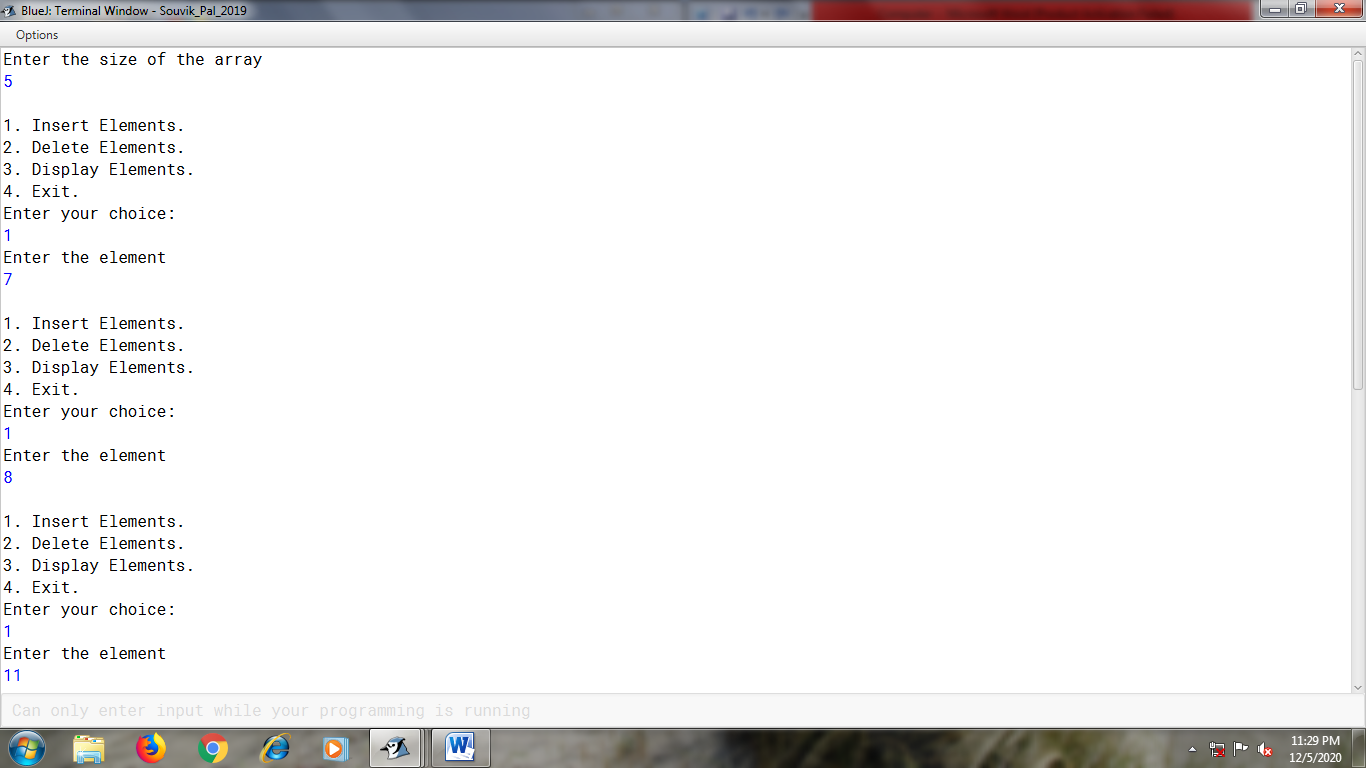
System.out.println(" "+queue[i]);

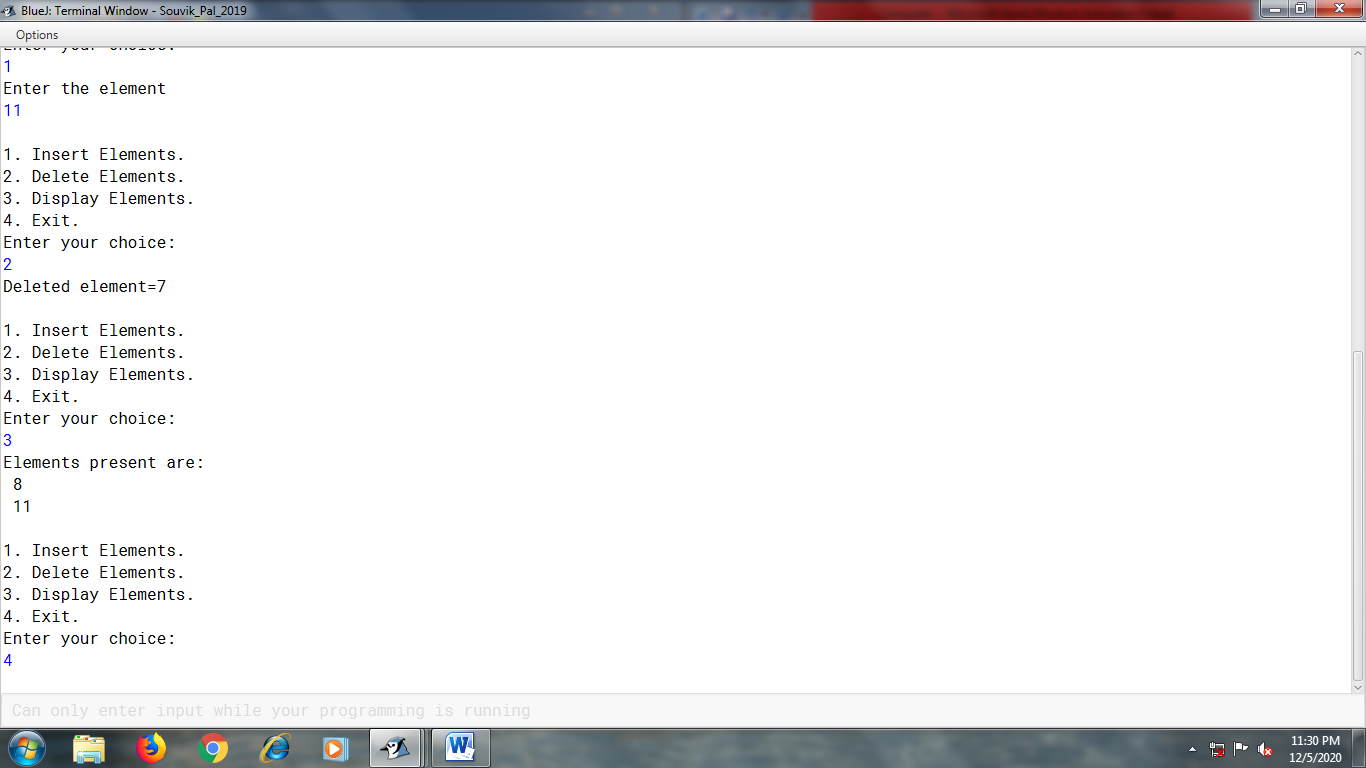
}

}

}

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:

