Ques 1 :

Ans :

#include<iostream>

using namespace std;

template<class sk>

sk linearSearch(sk arr[],sk size,sk num)

{

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

{

if(arr[i]==num)

return i;

}

return -1;

}

template<class sk>

sk binarySearch(sk arr[],sk size,sk num)

{

int start=0;

int last =size-1;

while(start<=last)

{

int mid=(start+last)/2;

if(num==arr[mid])

return mid;

else if(num>arr[mid])

start=mid+1;

else

last=mid-1;

}

return -1;

}

int main()

{

int x;

double size,num,index;

double arr[100];

char ch;

do

{

cout<<"ENTER"<<endl;

cout<<"1. LINEAR SEARCH"<<endl<<"2. BINARY SEARCH"<<endl;

cin>>x;

switch(x)

{

case 1 :

cout<<"Enter the size of array....."<<endl;

cin>>size;

cout<<"Enter the elements of array.... "<<endl;

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

cin>>arr[i];

cout<<"Enter the element to be searched for..."<<endl;

cin>>num;

index=linearSearch(arr,size,num);

if(index==-1)

cout<<"NO SUCH ELEMENT FOUND....";

else

cout<<"NUMBER FOUND AT "<<index+1<<" POSITION ...."<<endl;

cout<<"Enter Y to continue and N to exit..."<<endl;

cin>>ch;

break;

case 2 :

cout<<"Enter the size of array....."<<endl;

cin>>size;

cout<<"Enter the elements of array(in ascending order).... "<<endl;

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

cin>>arr[i];

cout<<"Enter the element to be searched for..."<<endl;

cin>>num;

index=binarySearch(arr,size,num);

if(index==-1)

cout<<"NO SUCH ELEMENT FOUND....";

else

cout<<"NUMBER FOUND AT "<<index+1<<" POSITION ...."<<endl;

cout<<"Enter Y to continue and N to exit..."<<endl;

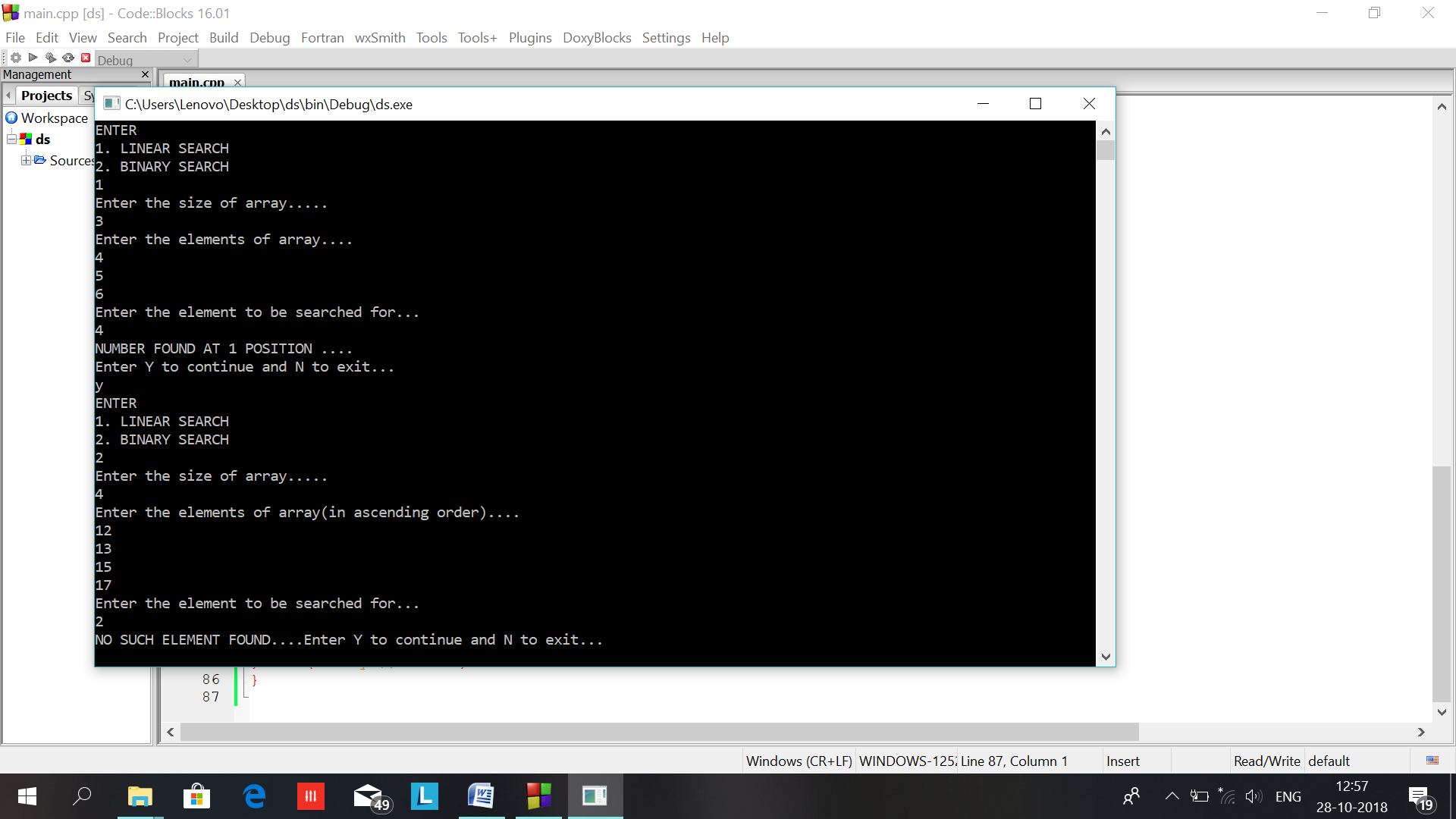
cin>>ch;

break;

}

}while(ch=='y'||ch=='Y');

}



Ques:2 //linked list

Ans:

#include<iostream>

using namespace std;

template<class t>

class node

{

public:

t data;

node<t> \*next;

};

template<class t1>

class list

{

int n;

node<int> \*first;

public:

list()

{

first=NULL;

}

void create()

{

char ch;

node<int> \*current,\*temp;

cout<<"Enter data:\n";

first=new node<int>;

cin>>first->data;

first->next=NULL;

current=first;

do

{

cout<<"Want to enter more data:\n";

cin>>ch;

if(ch=='y')

{

n=count();

this->insert(n+1);

}

}while(ch=='y');

}

void create\_beg()

{

cout<<"Enter how many nodes u want to enter in linked list:\n";

cin>>n;

node<int> \*temp;

first=new node<int>;

cout<<"Enter first node data:\n";

cin>>first->data;

first->next=NULL;

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

{

temp=new node<int>;

cout<<"Enter node data:\n";

cin>>temp->data;

temp->next=first;

first=temp;

}

}

void insert(t1 n)

{

int b=count();

if(n<=b+1)

{

node<int> \*current,\*temp;

current=first;

temp=new node<int>;

cout<<"Enter data:\n";

cin>>temp->data;

temp->next=NULL;

if(n==1)

{

temp->next=first;

first=temp;

}

else

{

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

current=current->next;

temp->next=current->next;

current->next=temp;

}

}

else

cout<<"Can't be inserted\n";

}

//search function

void search()

{

int flag=0;

cout<<"Enter data to be searched:\n";

cin>>n;

node<int> \*current,\*prev,\*temp;

int b=count();

current=first;

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

{

if(current->data==n)

{

flag=1;

break;

}

prev=current;

current=current->next;

}

if(flag==1)

{

int c;

cout<<"Data found:\nEnter what you wannna do:\n 1.delete data\n 2.replace it\n 3.do nothing\n";

cin>>c;

switch(c)

{

case 1:temp=current;

prev->next=current->next;

delete(temp);

cout<<"Data deleted:\n";

break;

case 2:cout<<"Enter new data:\n";

cin>>current->data;

cout<<"data replaced:\n";

break;

case 3:break;

default:cout<<"wrong choice:\n";

}

}

else

cout<<"Data not found:\n";

}

//overloading + operator

list\* operator +(list l1)

{

list l2;

l2.first=first;

node<int> \*current;

current=l2.first;

while(current->next!=NULL)

current=current->next;

current->next=l1.first;

return this;

}

//reverse function

void reverse()

{

node<int> \*a,\*b,\*temp;

a=first;

b=a->next;

temp=b->next;

a->next=NULL;

while(temp!=NULL)

{

b->next=a;

a=b;

b=temp;

temp=temp->next;

}

b->next=a;

first=b;

}

//counting nodes in a linked list

int count()

{

node<int> \*current;

int c=0;

current=first;

while(current!=NULL)

{

c++;

current=current->next;

}

return c;

}

//function to display linked list

void display()

{

//getchar();

node<int> \*current;

current=first;

cout<<"The data in linked list:\n";

while(current!=NULL)

{

cout<<current->data<<" -> ";

current=current->next;

}

cout<<endl;

}

};

int main()

{

//initial declarations

int cho,n;

char ch;

list<int> l1,l2,\*l3;

//creating linked list and working on them

cout<<"Want to create linked list:\n1.Simple\n2.from beginning:\n";

cin>>cho;

switch(cho)

{

case 1:l1.create();

break;

case 2:l1.create\_beg();

break;

default:cout<<"Wrong choice:\n";

}

if((cho==1)||(cho==2))

{

l1.display();

//doing insertion

do

{

cout<<"Want to insert a node:\n(y\n)";

cin>>ch;

if(ch=='y')

{

cout<<"Enter the position of insertion;\n";

cin>>n;

l1.insert(n);

}

}while(ch=='y');

cout<<"The linked list after all insertions:\n";

l1.display();

//doing searching,deleting,replacing

do

{

cout<<"Want to search a data:\n";

cin>>ch;

if(ch=='y')

l1.search();

}while(ch=='y');

cout<<"The linked list after searching and as so:\n";

l1.display();

//creating new linked list

cout<<"Want to create new linked list:\n";

char cht;

cin>>cht;

if(cht=='y')

{

l2.create();

l2.display();

//concatenating strings

cout<<"Want to concatenate two linked lists:\n";

cin>>ch;

if(ch=='y')

{

l3=l1+l2;

cout<<"new linked list after concatenation:\n";

l3->display();

}

}

//reversing linked lists

cout<<"Want to reverse the linked list:\n";

cin>>ch;

if(ch=='y')

{

cout<<"Which linked list\n1.";

l1.display();

if(cht=='y')

{

cout<<"\n2.";

l2.display();

cout<<"\n3.";

l3->display();

}

cin>>cho;

//fflush(stdin);

if(((cho>=1)&&(cht=='y'))||((cho==1)&&(cht!='y')))

{

switch(cho)

{

case 1:l1.reverse();

cout<<"\nnew linked list after reversing:\n";

l1.display();

break;

case 2:l2.reverse();

cout<<"\nnew linked list after reversing:\n";

l2.display();

break;

case 3:l3->reverse();

cout<<"\nnew linked list after reversing:\n";

l3->display();

break;

default:cout<<"Wrong choice entered:\n";

}

}

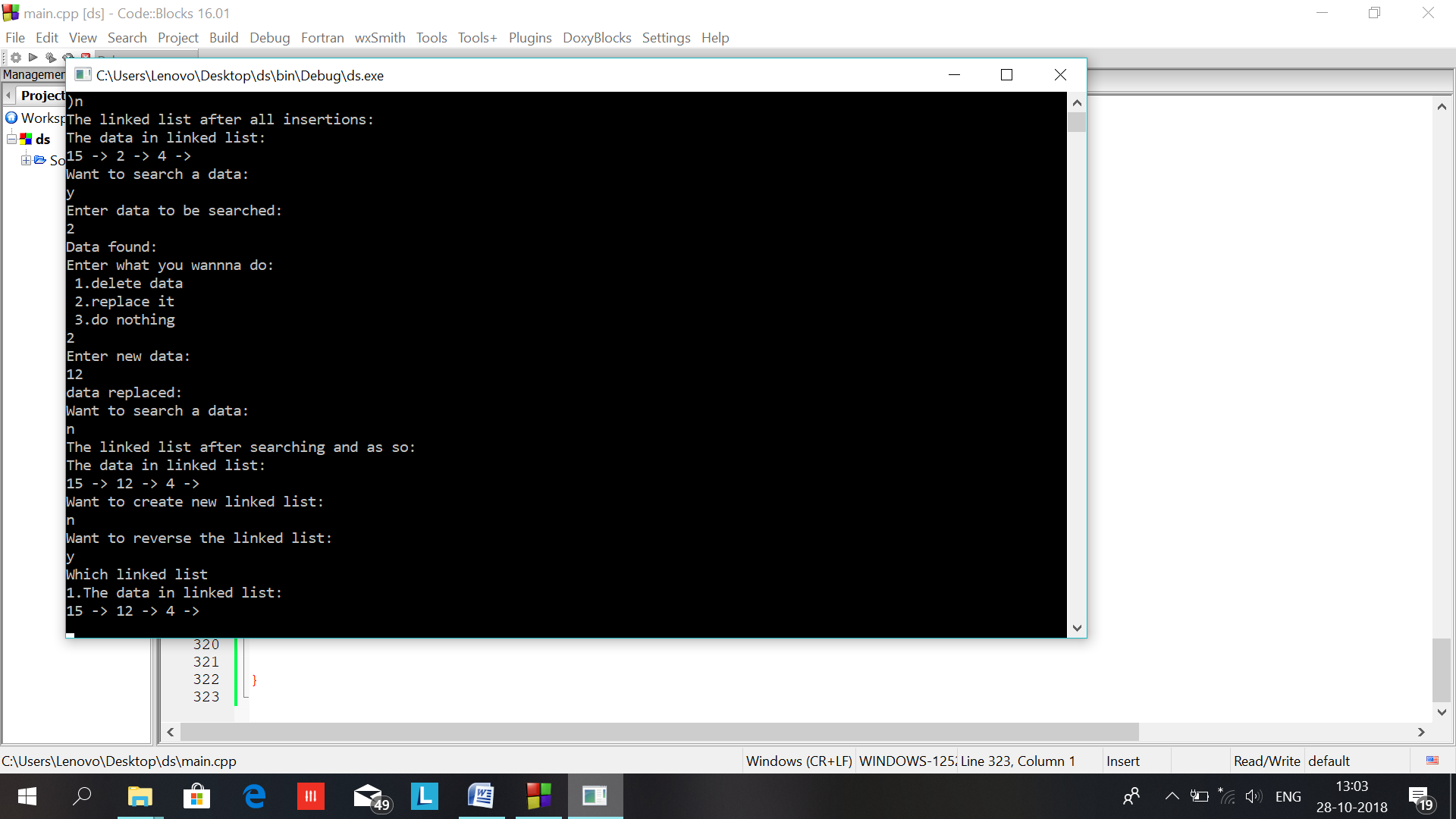
else

cout<<"Wrong choice entered:\n";

}

}

}



Ques:3 //sort

Ans:

#include<iostream>

#include<limits.h>

using namespace std;

template<class sk>

void selectionSort(sk arr[],sk size)

{

int temp,pos,small;

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

{

small=arr[i];

pos=i;

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

{

if(arr[j]<small)

{

small=arr[j];pos=j;

}

}

temp=arr[i];

arr[i]=arr[pos];

arr[pos]=temp;

cout<<endl<<"array after pass -- "<<i+1<<" -is ";

for(int j=0;j<size;j++) cout<<arr[j]<<" ";

}

}

template<class sk>

void bubbleSort(sk arr[],sk size)

{

sk temp;

int c=0;

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

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

{

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

{

temp = arr[i];

arr[i]=arr[j];

arr[j]=temp;

}

}

cout<<endl<<"array after pass -- "<<i+1<<" -is ";

for(int m=0;m<size;m++) cout<<arr[m]<<" ";

}

}

template<class sk>

void insertionSort(sk arr[],sk size)

{

int tmp,j;

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

{

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

{

if(arr[j]<arr[j-1])

{

tmp=arr[j];

arr[j]=arr[j-1];

arr[j-1]=tmp;

}

}

cout<<endl<<"array after pass -- "<<i+1<<" -is ";

for(int m=0;m<size;m++) cout<<arr[m]<<" ";

}

}

int main()

{

int x;

double size,num,index;

double arr[100];

char ch;

do

{

cout<<"ENTER"<<endl;

cout<<"1. SELECTION SORT"<<endl<<"2. BUBBLE SORT"<<endl<<"3. INSERTION SORT"<<endl;

cin>>x;

switch(x)

{

case 1 :

cout<<"Enter the size of array....."<<endl;

cin>>size;

cout<<"Enter the elements of array.... "<<endl;

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

cin>>arr[i];

selectionSort(arr,size);

cout<<endl<<"THE SORTED ARRAY AFTER USING SELECTION SORT IS : " <<endl;

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

cout<<arr[i]<<" ";

cout<<endl<<"Enter Y to continue and N to exit..."<<endl;

cin>>ch;

break;

case 2 :

cout<<"Enter the size of array....."<<endl;

cin>>size;

cout<<"Enter the elements of array.... "<<endl;

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

cin>>arr[i];

bubbleSort(arr,size);

cout<<"THE SORTED ARRAY AFTER USING BUBBLE SORT IS : " <<endl;

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

cout<<arr[k]<<" ";

cout<<endl<<"Enter Y to continue and N to exit..."<<endl;

cin>>ch;

break;

case 3 :

cout<<"Enter the size of array....."<<endl;

cin>>size;

cout<<"Enter the elements of array.... "<<endl;

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

cin>>arr[i];

insertionSort(arr,size);

cout<<endl<<"THE SORTED ARRAY AFTER USING INSERTION SORT IS : " <<endl;

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

cout<<arr[i]<<" ";

cout<<endl<<"Enter Y to continue and N to exit..."<<endl;

cin>>ch;

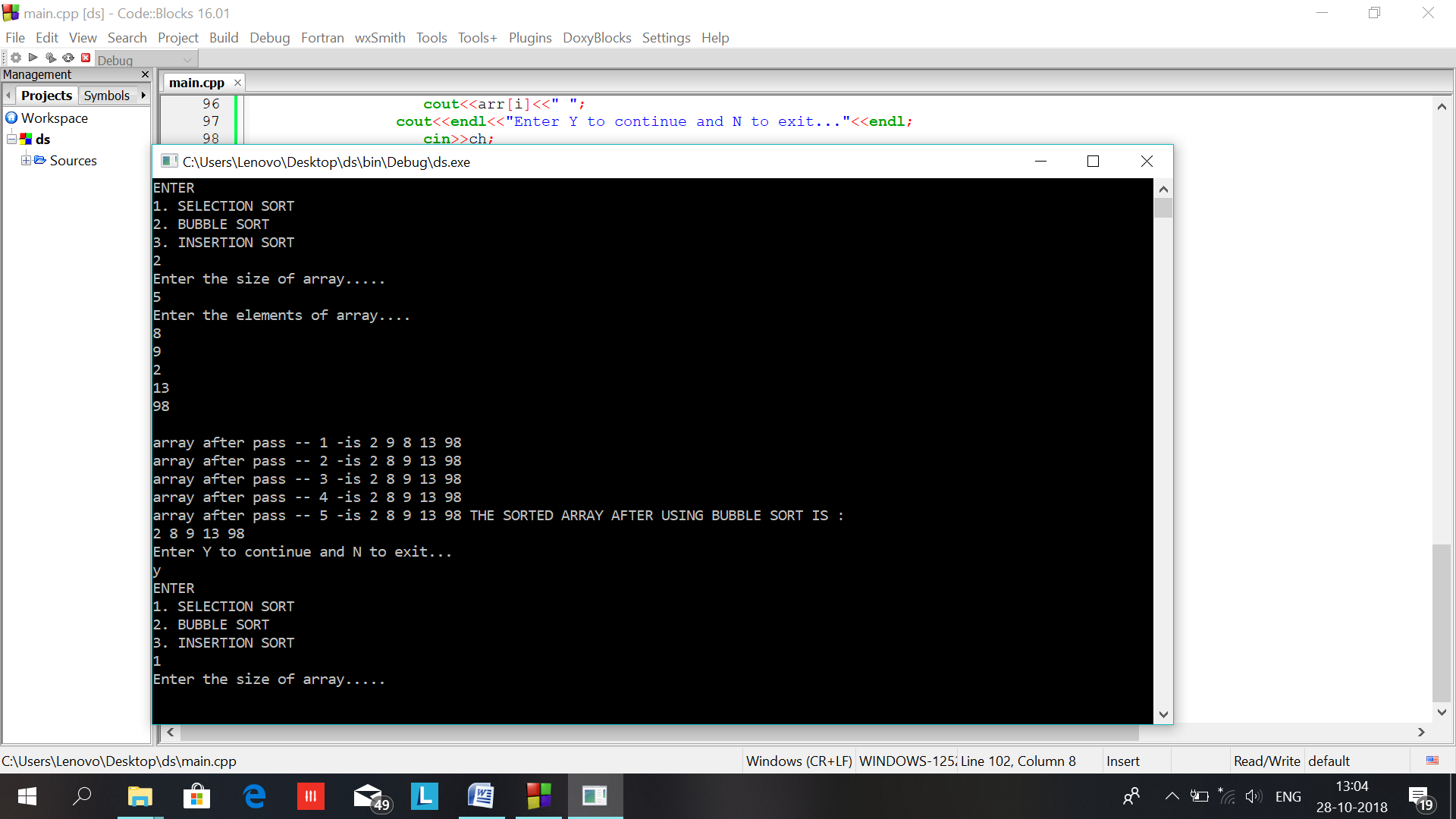
break;

}

}

while(ch=='y'||ch=='Y');

}



Ques:4 //doubly linked list

Ans:

#include<iostream>

using namespace std;

template<class t>

class node

{

public:

t data;

node \*prev,\*next;

};

template<class t>

class dlist

{

int n;

node<t> \*first,\*last;

public:

dlist()

{

first=NULL;

last=NULL;

}

//create function

void create()

{

node<t> \*current,\*temp;

char ch;

//fflush(stdin);

first=new node<t>;

cout<<"Enter data for first node:\n";

cin>>first->data;

current=first;

first->next=NULL;

first->prev=NULL;

last=first;

do

{

cout<<"Want to enter more data:\n";

cin>>ch;

if(ch=='y')

{

n=count();

this->insert(n+1);

}

}while(ch=='y');

}

//display function

void display()

{

node<t> \*current;

current=first;

cout<<"The data in linked list:\n";

while(current!=NULL)

{

cout<<current->data<<" <-> ";

current=current->next;

}

cout<<"\n";

}

//reverse function

void reverse()

{

n=count();

//fflush(stdin);

node<t> \*current;

current=last;

cout<<"The data after reversing the linked list:\n";

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

{

cout<<current->data<<" -> ";

current=current->prev;

}

}

//count function

int count()

{

int c=0;

node<t> \*current;

current=first;

while(current!=NULL)

{

c++;

current=current->next;

}

return c;

}

//insert function

void insert(int n1)

{

int b=count();

if(n1<=b+1)

{

node<t> \*current,\*forward,\*temp;

current=first;

temp=new node<t>;

cout<<"Enter data:\n";

cin>>temp->data;

temp->next=temp->prev=NULL;

if(n1==1)

{

temp->next=first;

first->prev=temp;

first=temp;

}

else if(n1<=b)

{

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

current=current->next;

forward=current->next;

temp->next=forward;

current->next=temp;

temp->prev=current;

forward->prev=temp;

}

else

{

last->next=temp;

temp->prev=last;

last=temp;

}

}

else

cout<<"Can't be inserted\n";

}

//search function

void search()

{

int flag=0;

cout<<"Enter data to be searched:\n";

cin>>n;

node<t> \*current,\*previ,\*temp;

int b=count();

current=first;

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

{

if(current->data==n)

{

flag=1;

break;

}

current=current->next;

}

if(flag!=0)

{

previ=current->prev;

int c;

cout<<"Data found:\nEnter what you wannna do:\n 1.delete data\n 2.replace it\n 3.do nothing\n";

cin>>c;

switch(c)

{

case 1:temp=current;

if(current->next!=NULL)

{

current=current->next;

previ->next=current;

current->prev=previ;

}

else

{

previ->next=NULL;

current->prev=NULL;

}

delete(temp);

cout<<"Data deleted:\n";

this->display();

break;

case 2:cout<<"Enter new data:\n";

cin>>current->data;

cout<<"data replaced:\n";

this->display();

break;

case 3:break;

default:cout<<"wrong choice:\n";

}

//getchar();

}

else

cout<<"Data not found:\n";

}

//overloading + operator

dlist operator +(dlist l)

{

dlist l6;

l6.first=first;

l6.last=last;

l6.last->next=l.first;

l.first->prev=l6.last;

return l6;

}

};

int main()

{

int n;

char ch;

dlist<int> l1,l3,l2;

l1.create();

l1.display();

//doing insertion

do

{

cout<<"Want to insert a node:\n";

cin>>ch;

if(ch=='y')

{

cout<<"Enter the position of insertion;\n";

cin>>n;

l1.insert(n);

}

}while(ch=='y');

cout<<"The linked list after all insertions:\n";

l1.display();

//doing searching,deleting,replacing

do

{

cout<<"Want to search a data:\n";

cin>>ch;

if(ch=='y')

l1.search();

}while(ch=='y');

cout<<"The linked list after searching and as so:\n";

l1.display();

//creating new linked list

cout<<"Want to create new linked list:\n";

char cht;

cin>>cht;

if(cht=='y')

{

l2.create();

l2.display();

//concatenating strings

cout<<"Want to concatenate two linked lists:\n";

cin>>ch;

if(ch=='y')

{

l3=l1+l2;

cout<<"new linked list after concatenation:\n";

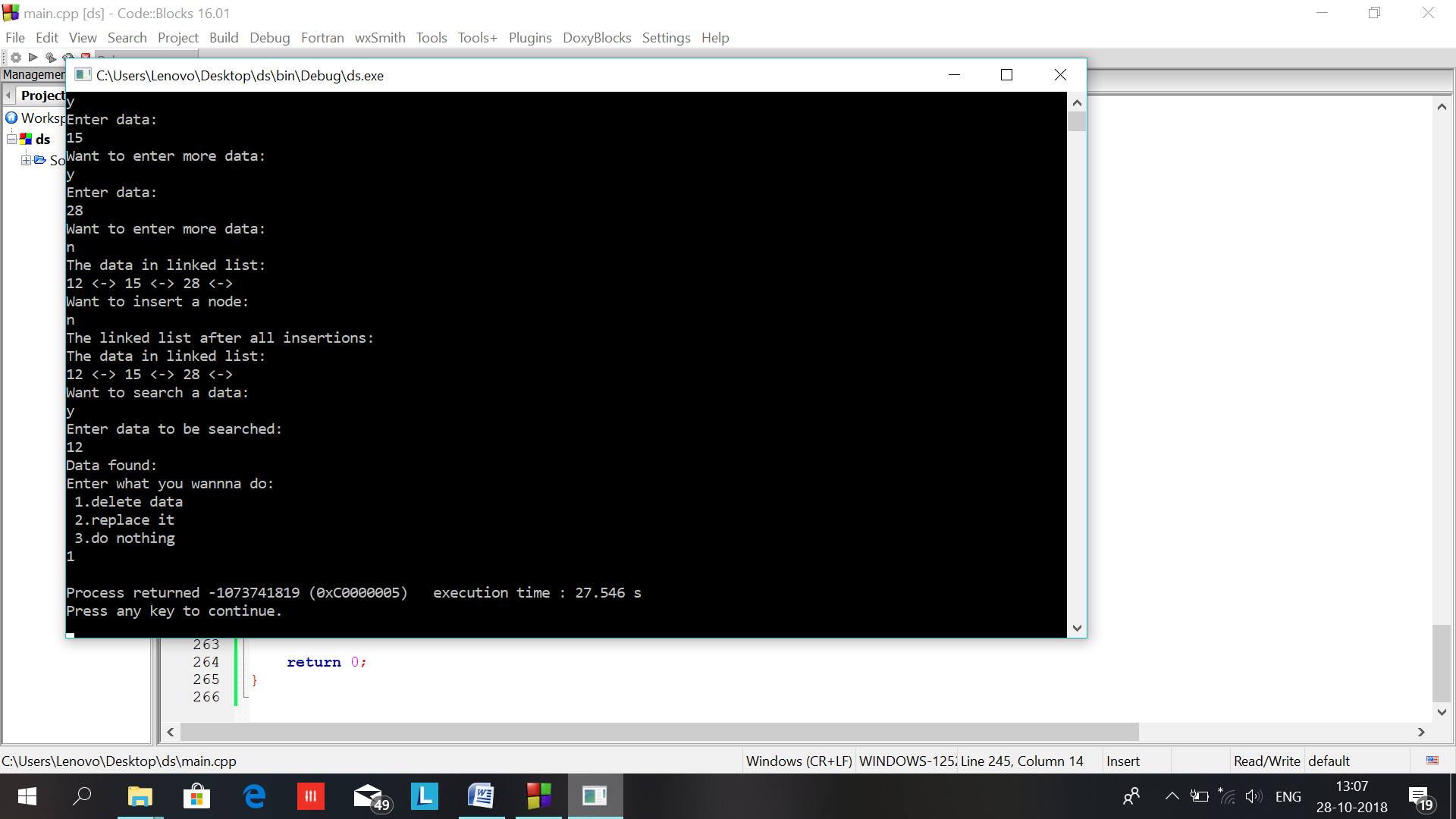
l3.display();

}

}

return 0;

}



Ques:5 circluar linked list

Ans:

#include<iostream>

using namespace std;

template<class t>

class node

{

public:

t data;

node \*next;

};

template<class t>

class clist

{

int n;

node<t> \*first,\*last;

public:

clist()

{

first=NULL;

}

//create function

void create()

{

node<t> \*current,\*temp;

cout<<"Enter how many nodes you want to enter in linked list:\n";

cin>>n;

//fflush(stdin);

if(n>0)

{

if(first==NULL)

{

first=new node<t>;

cout<<"Enter data for first node:\n";

cin>>first->data;

//fflush(stdin);

first->next=first;

}

current=first;

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

{

cout<<"Enter data:\n";

temp=new node<t>;

cin>>temp->data;

//fflush(stdin);

temp->next=current->next;

current->next=temp;

current=current->next;

}

last=current;

}

}

//count the list

int count()

{

node<t> \*current;

current=first;

int c=0;

while(current->next!=first)

{

c++;

current=current->next;

}

c++;

return c;

}

//insert function

void insert()

{

cout<<"Enter the position of insertion;\n";

cin>>n;

int b=count();

if(n<=b+1)

{

node<t> \*current,\*temp;

current=first;

temp=new node<t>;

cout<<"Enter data:\n";

cin>>temp->data;

temp->next=NULL;

if(n==1)

{

temp->next=first;

first=temp;

last->next=first;

}

else

{

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

current=current->next;

temp->next=current->next;

current->next=temp;

}

}

else

cout<<"Can't be inserted\n";

}

//deletion now vomes

void search()

{

int flag=0;

cout<<"Enter data to be searched:\n";

cin>>n;

node<t> \*current,\*prev,\*temp;

int b=count();

current=first;

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

{

if(current->data==n)

{

flag=1;

break;

}

prev=current;

current=current->next;

}

if(flag==1)

{

int c;

cout<<"Data found:\nEnter what you wannna do:\n 1.delete data\n 2.replace it\n 3.do nothing\n";

cin>>c;

switch(c)

{

case 1:temp=current;

prev->next=current->next;

delete(temp);

cout<<"Data deleted:\n";

break;

case 2:cout<<"Enter new data:\n";

cin>>current->data;

cout<<"data replaced:\n";

break;

case 3:break;

default:cout<<"wrong choice:\n";

}

}

else

cout<<"Data not found:\n";

}

//reverse function

void reverse()

{

node<t> \*a,\*b,\*temp;

a=first;

b=a->next;

temp=b->next;

a->next=NULL;

while(temp!=first)

{

//fflush(stdin);

b->next=a;

a=b;

b=temp;

temp=temp->next;

//this->display();

}

b->next=a;

first->next=b;

first=first->next;

}

//create display

void display()

{

node<t> \*current;

current=first;

while(current->next!=first)

{

cout<<current->data<<" -> ";

current=current->next;

}

cout<<current->data<<" -> \n";

}

};

int main()

{

char ch;

clist<int> l1;

l1.create();

l1.display();

//doing insertion

do

{

cout<<"Want to insert a node:\n";

cin>>ch;

if(ch=='y')

l1.insert();

}while(ch=='y');

cout<<"The linked list after all insertions:\n";

l1.display();

do

{

cout<<"Want to search a node:\n";

cin>>ch;

if(ch=='y')

l1.search();

}while(ch=='y');

cout<<"The linked list after all operations:\n";

l1.display();

cout<<"Want to see reversed linked list:\n";

cin>>ch;

if(ch=='y')

{

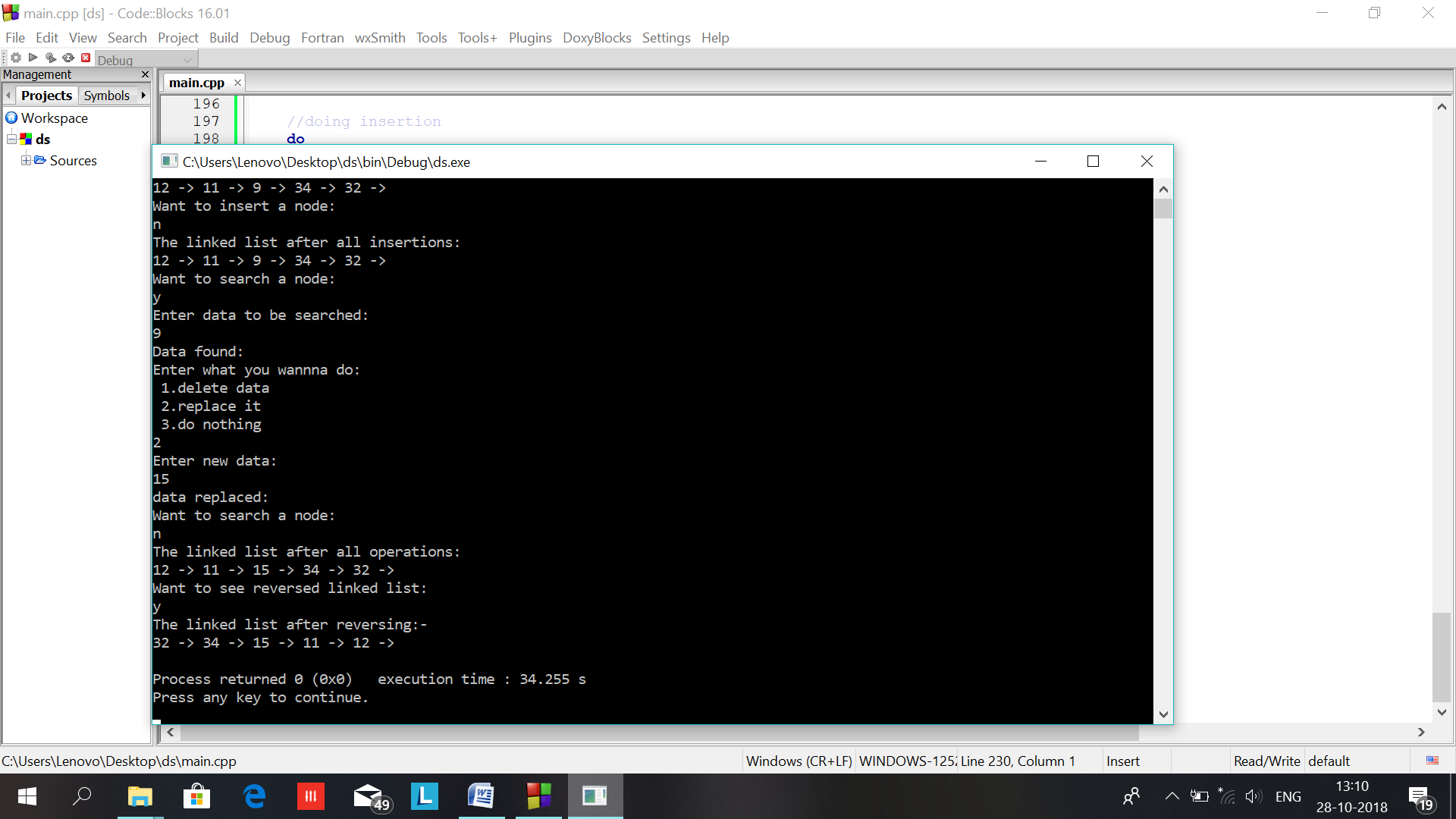
l1.reverse();

}

cout<<"The linked list after reversing:-\n";

l1.display();

}



Ques:6 stack using linked list

Ans:

#include<iostream>

using namespace std;

template<class t>

//const int size=5;

class stack

{

public:

t data;

stack \*prev;

};

template<class t>

class stacks

{

//int stack[size];

stack<t> \*top;

public:

stacks()

{

top=NULL;

}

void push(t n)

{

stack<t> \*temp;

temp=new stack<t>;

temp->data=n;

temp->prev=top;

top=temp;

}

t pop()

{

t a;

if(top==NULL)

{

cout<<"underflow:\n";

return -1;

}

else

{

a=top->data;

top=top->prev;

return a;

}

}

void display()

{

cout<<"The data in stack is:\n";

while(top!=NULL)

{

cout<<top->data<<"\n^\n"<<endl;

top=top->prev;

}

}

int empty()

{

if(top==NULL)

return -1;

else

return 1;

}

void clear(){top=-1;}

};

int main()

{

char ch;

int n,n1;

stacks<int> s1;

do

{

cout<<"Enter if u want to push any element:\n";

cin>>ch;

if(ch=='y')

{

cout<<"Enter the element to be pushed:\n";

cin>>n;

s1.push(n);

}

}while(ch=='y');

do

{

cout<<"Enter if u want to pop any element:\n";

cin>>ch;

if(ch=='y')

{

n=s1.pop();

if(n==-1)

break;

cout<<"Popped element is : "<<n<<"\n";

}

}while(ch=='y');

if(s1.empty()==1)

{

cout<<"The stack after all operations:\n";

s1.display();

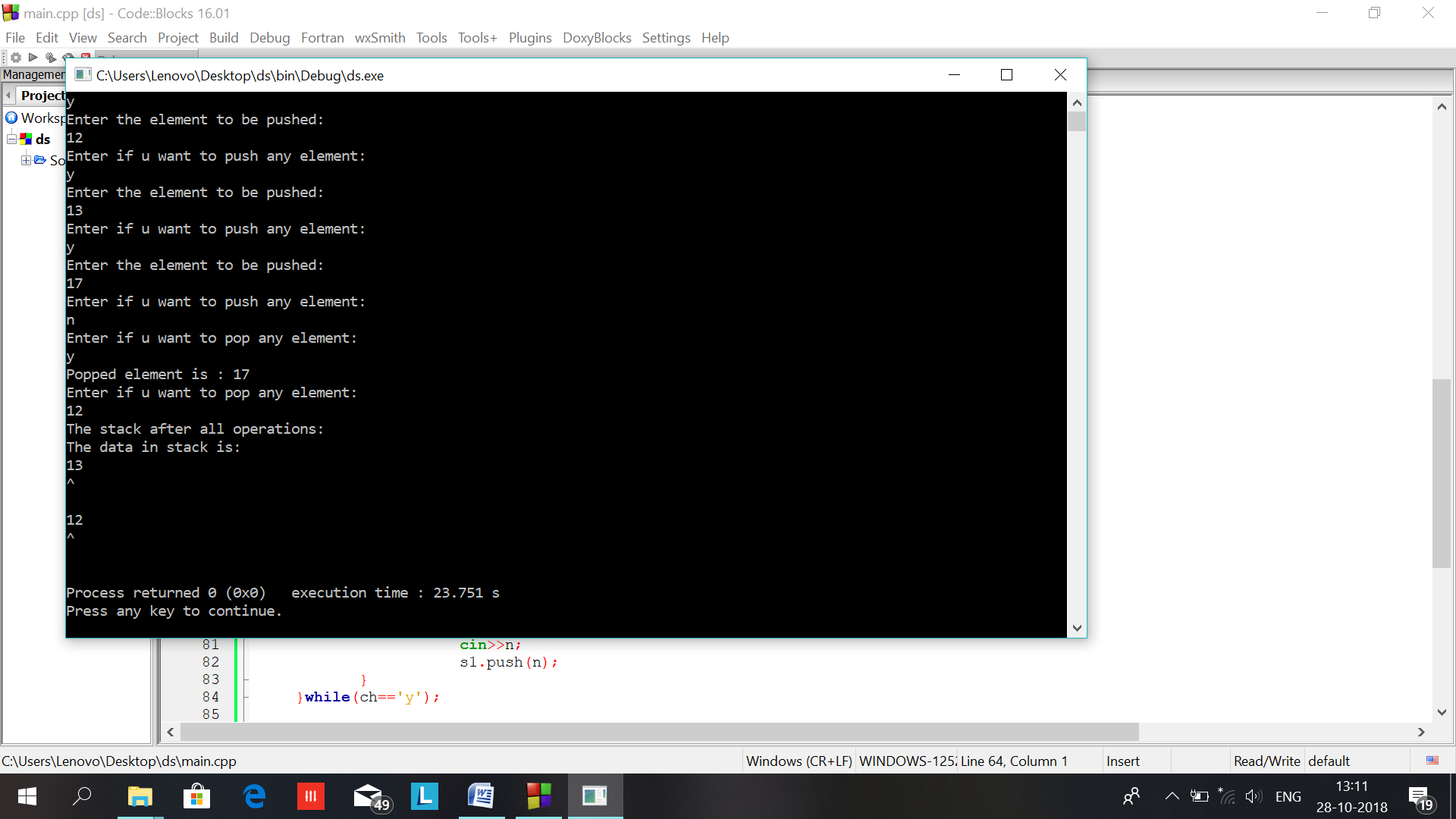
}

else

cout<<"NULL stack\n";

return 0;

}



Ques:7 stack using array

Ans:

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

template<class t>

class stack

{

t stk[10];

t top;

public:

stack()

{

top=-1;

}

void push(int x)

{

if(top > 10)

{

cout <<"stack over flow";

return;

}

stk[++top]=x;

cout <<" Inserted : " <<x<<endl;

}

void pop()

{

if(top <0)

{

cout <<"stack under flow";

return;

}

cout <<" \n Deleted : " <<stk[top--];

}

void display()

{

if(top<0)

{

cout <<" stack empty";

return;

}

for(int i=top;i>=0;i--)

cout <<stk[i] <<" ";

}

};

main()

{

int ch;

stack<int> st;

while(1)

{

cout <<"\n 1.push 2.pop 3.display 4.exit \nEnter ur choice \n";

cin >> ch;

switch(ch)

{

case 1: cout <<"\n enter the element \n ";

cin >> ch;

st.push(ch);

break;

case 2: st.pop(); break;

case 3: st.display();break;

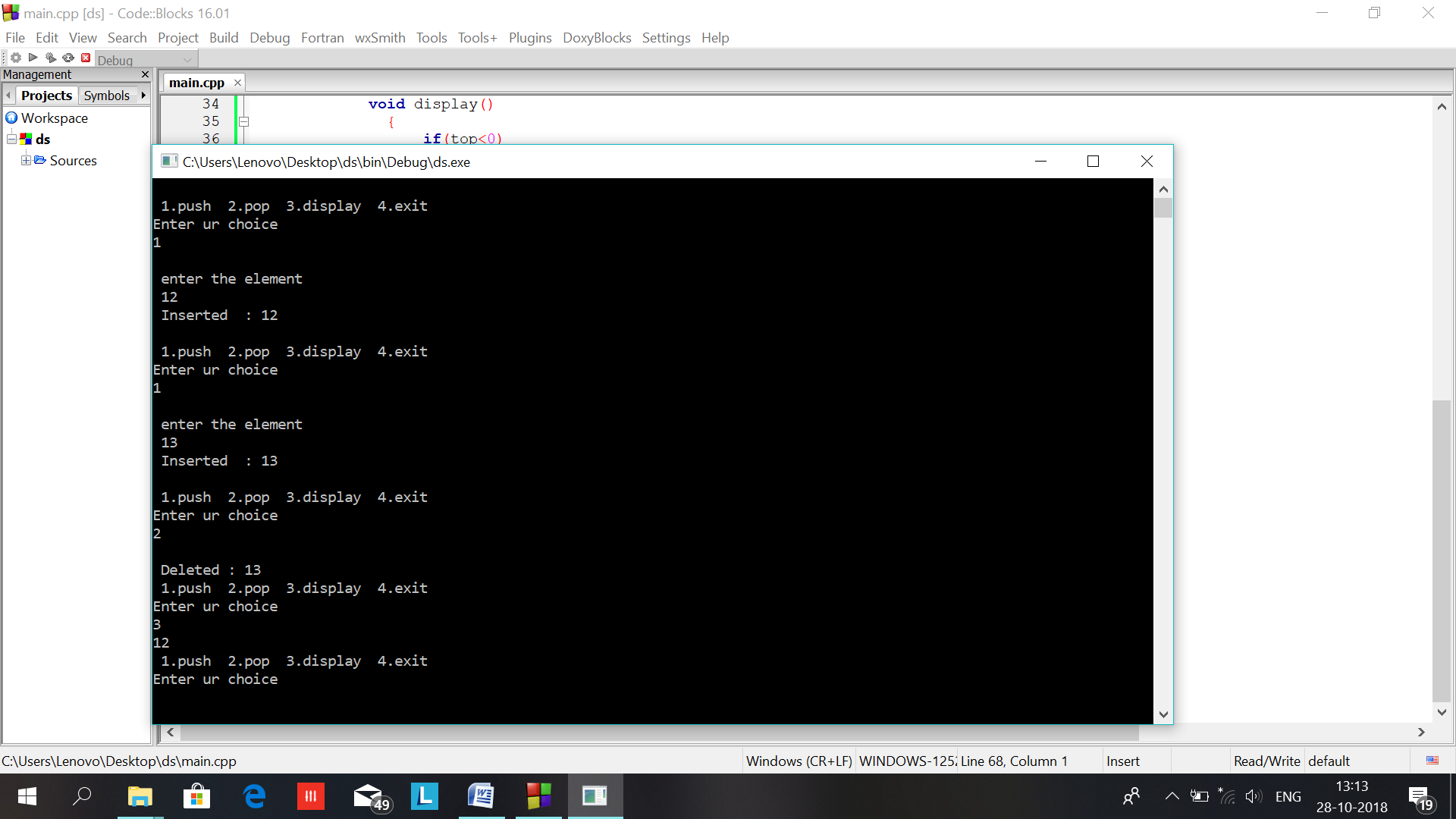
case 4: exit(0);

}

}

return (0);

}



Ques:8 circluar queue

Ans:

#include<iostream>

using namespace std;

const int size=5;

int flag=0;

template<class t>

class CQ

{

t queue[size];

int front,rear;

public:

CQ()

{

front=rear=-1;

}

void insert(t n)

{

if((rear+1)%size!=front)

{

rear=(rear+1)%size;

queue[rear]=n;

if(front<0)

front=0;

}

else

{

cout<<"Queue overflow element not inserted:\n";

flag=-1;

//fflush(stdin);

}

}

t remove()

{

t a;

if(empty()!=-1)

{

if((front==rear)&&(front>=0))

{

a=queue[front];

front=rear=-1;

}

else if(front!=size-1)

{

a=queue[front++];

}

else

{

front=0;

a=queue[front];

}

return a;

}

else

{

cout<<"Queue underflow:\n";

return -1;

}

}

int empty()

{

if(front==-1)

return -1;

else

return 0;

}

void display()

{

if(empty()==-1)

{

cout<<"Empty queue:\n";

}

else

{

int r,f;

r=rear;

f=front;

while((f+1)%size!=(r+1)%size)

{

cout<<queue[f]<<endl;

f=(f+1)%size;

}

cout<<queue[f]<<endl;

/\*cout<<"the values in the original array:\n";

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

{

cout<<queue[i]<<" ";

}

cout<<endl;\*/

}

}

};

int main()

{

CQ<char> q1;

char ch;

char n;

flag=0;

do

{

cout<<"Enter 'y' if want to enter any element:\n";

cin>>ch;

if(ch=='y')

{

cout<<"Enter the element:\n";

cin>>n;

q1.insert(n);

if(flag==-1)

break;

}

}while(ch=='y');

cout<<"The resulting queue:\n";

q1.display();

flag=0;

if(q1.empty()!=-1)

{

do

{

cout<<"Enter 'y' if want to remove any element:\n";

cin>>ch;

if(ch=='y')

{

n=q1.remove();

if(n==-1)

break;

}

}while(ch=='y');

cout<<"The resulting queue:\n";

q1.display();

do

{

cout<<"Enter 'y' if want to enter any element:\n";

cin>>ch;

if(ch=='y')

{

cout<<"Enter the element:\n";

cin>>n;

q1.insert(n);

if(flag==-1)

break;

}

}while(ch=='y');

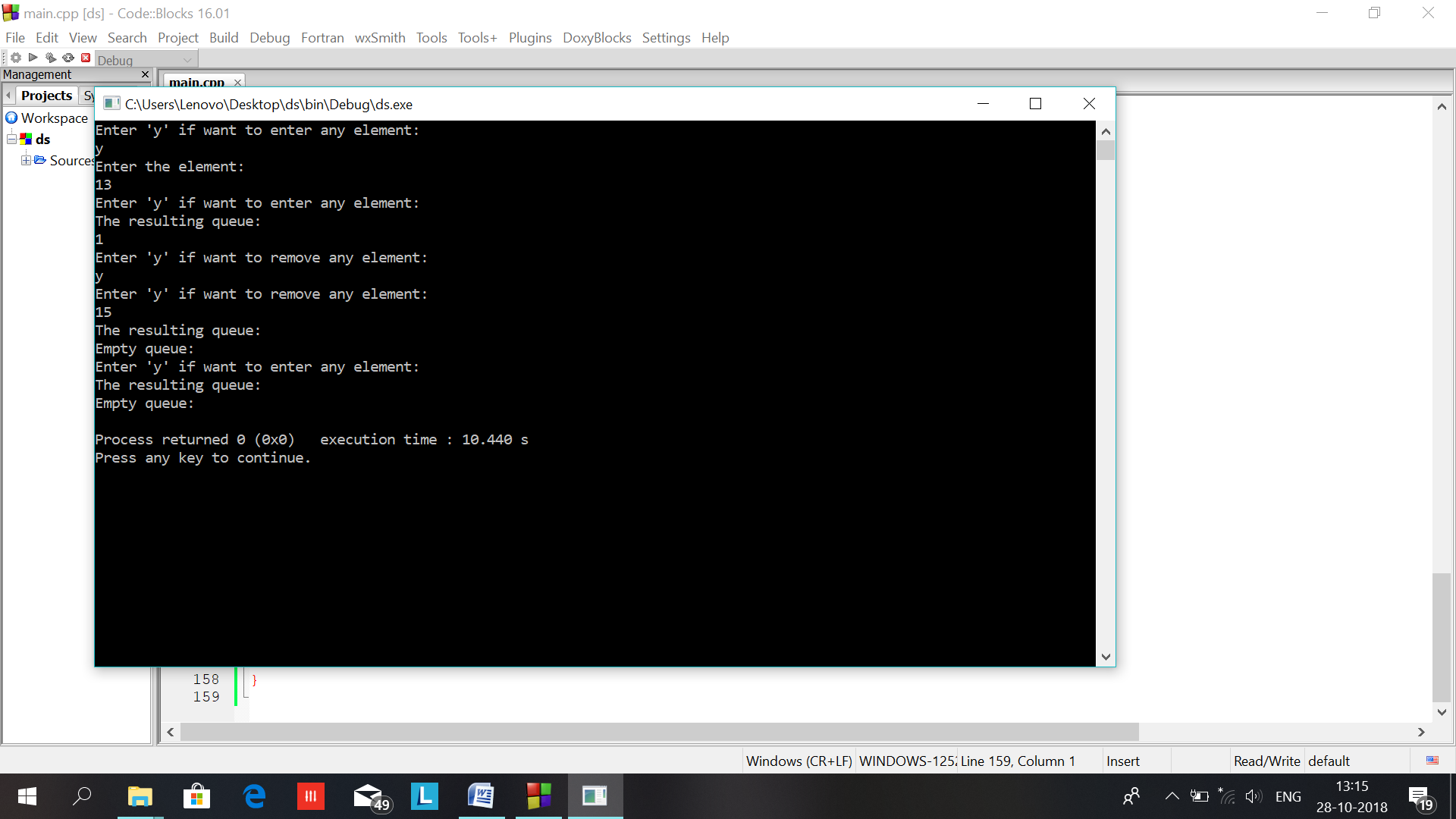
cout<<"The resulting queue:\n";

q1.display();

}

return 0;

}



#include<iostream>

using namespace std;

const int size=5;

int flag=0;

template<class t>

class CQ

{

t queue[size];

int front,rear;

public:

CQ()

{

front=rear=-1;

}

void insert(t n)

{

if((rear+1)%size!=front)

{

rear=(rear+1)%size;

queue[rear]=n;

if(front<0)

front=0;

}

else

{

cout<<"Queue overflow element not inserted:\n";

flag=-1;

//fflush(stdin);

}

}

t remove()

{

t a;

if(empty()!=-1)

{

if((front==rear)&&(front>=0))

{

a=queue[front];

front=rear=-1;

}

else if(front!=size-1)

{

a=queue[front++];

}

else

{

front=0;

a=queue[front];

}

return a;

}

else

{

cout<<"Queue underflow:\n";

return -1;

}

}

int empty()

{

if(front==-1)

return -1;

else

return 0;

}

void display()

{

if(empty()==-1)

{

cout<<"Empty queue:\n";

}

else

{

int r,f;

r=rear;

f=front;

while((f+1)%size!=(r+1)%size)

{

cout<<queue[f]<<endl;

f=(f+1)%size;

}

cout<<queue[f]<<endl;

/\*cout<<"the values in the original array:\n";

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

{

cout<<queue[i]<<" ";

}

cout<<endl;\*/

}

}

};

int main()

{

CQ<char> q1;

char ch;

char n;

flag=0;

do

{

cout<<"Enter 'y' if want to enter any element:\n";

cin>>ch;

if(ch=='y')

{

cout<<"Enter the element:\n";

cin>>n;

q1.insert(n);

if(flag==-1)

break;

}

}while(ch=='y');

cout<<"The resulting queue:\n";

q1.display();

flag=0;

if(q1.empty()!=-1)

{

do

{

cout<<"Enter 'y' if want to remove any element:\n";

cin>>ch;

if(ch=='y')

{

n=q1.remove();

if(n==-1)

break;

}

}while(ch=='y');

cout<<"The resulting queue:\n";

q1.display();

do

{

cout<<"Enter 'y' if want to enter any element:\n";

cin>>ch;

if(ch=='y')

{

cout<<"Enter the element:\n";

cin>>n;

q1.insert(n);

if(flag==-1)

break;

}

}while(ch=='y');

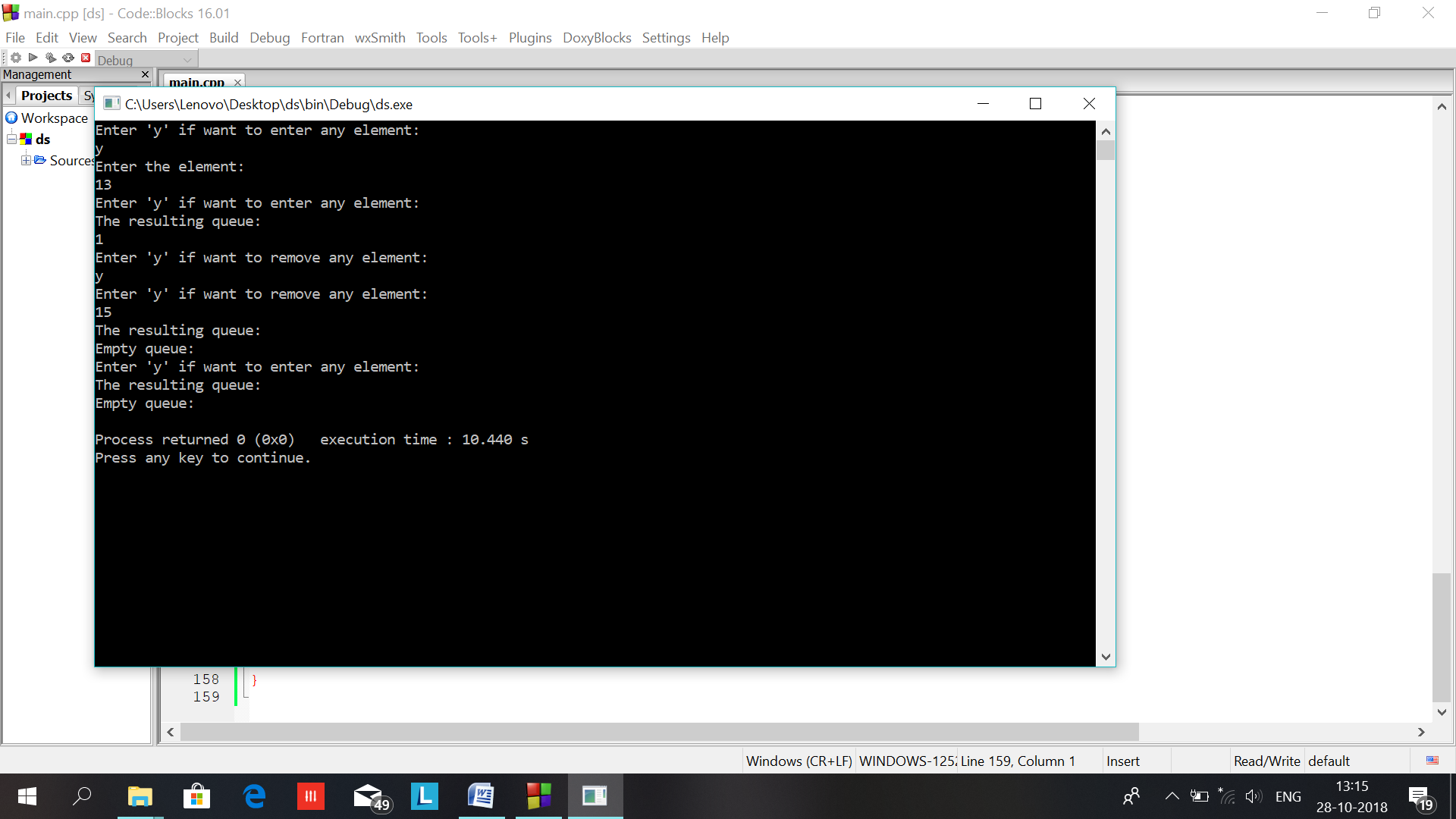
cout<<"The resulting queue:\n";

q1.display();

}

return 0;

}



Ques:9 //double ended queue

Ans:

#include<iostream>

#include<cstdlib>

using namespace std;

struct node{

int info;

struct node \*next;

};

class Queue{

private:

node \*rear;

node \*front;

public:

Queue();

void enqueue();

void dequeue();

void display();

};

Queue::Queue(){

rear = NULL;

front = NULL;

}

void Queue::enqueue(){

int data;

node \*temp = new node;

cout<<"\nEnter the data to enqueue: ";

cin>>data;

temp->info = data;

temp->next = NULL;

if(front == NULL){

front = temp;

}else{

rear->next = temp;

}

rear = temp;

}

void Queue::dequeue(){

node \*temp = new node;

if(front == NULL){

cout<<"\nQueue is Emtpty\n";

}else{

temp = front;

front = front->next;

cout<<"\n The data Dequeued is "<<temp->info;

delete temp;

}

}

void Queue::display(){

node \*p = new node;

p = front;

if(front == NULL){

cout<<"\nNothing to Display\n";

}else{

cout<<" \n the elements in queue are: ";

while(p!=NULL){

cout<<endl<<p->info;

p = p->next;

}

}

}

int main(){

Queue queue;

int choice;

while(true){

cout<<"\n1.Enqueue\n2. Dequeue\n3. Display\n 4.Quit";

cout<<"\nEnter your choice: ";

cin>>choice;

switch(choice){

case 1:

queue.enqueue();

break;

case 2:

queue.dequeue();

break;

case 3:

queue.display();

break;

case 4:

exit(0);

break;

default:

cout<<"\nInvalid Input. Try again! \n";

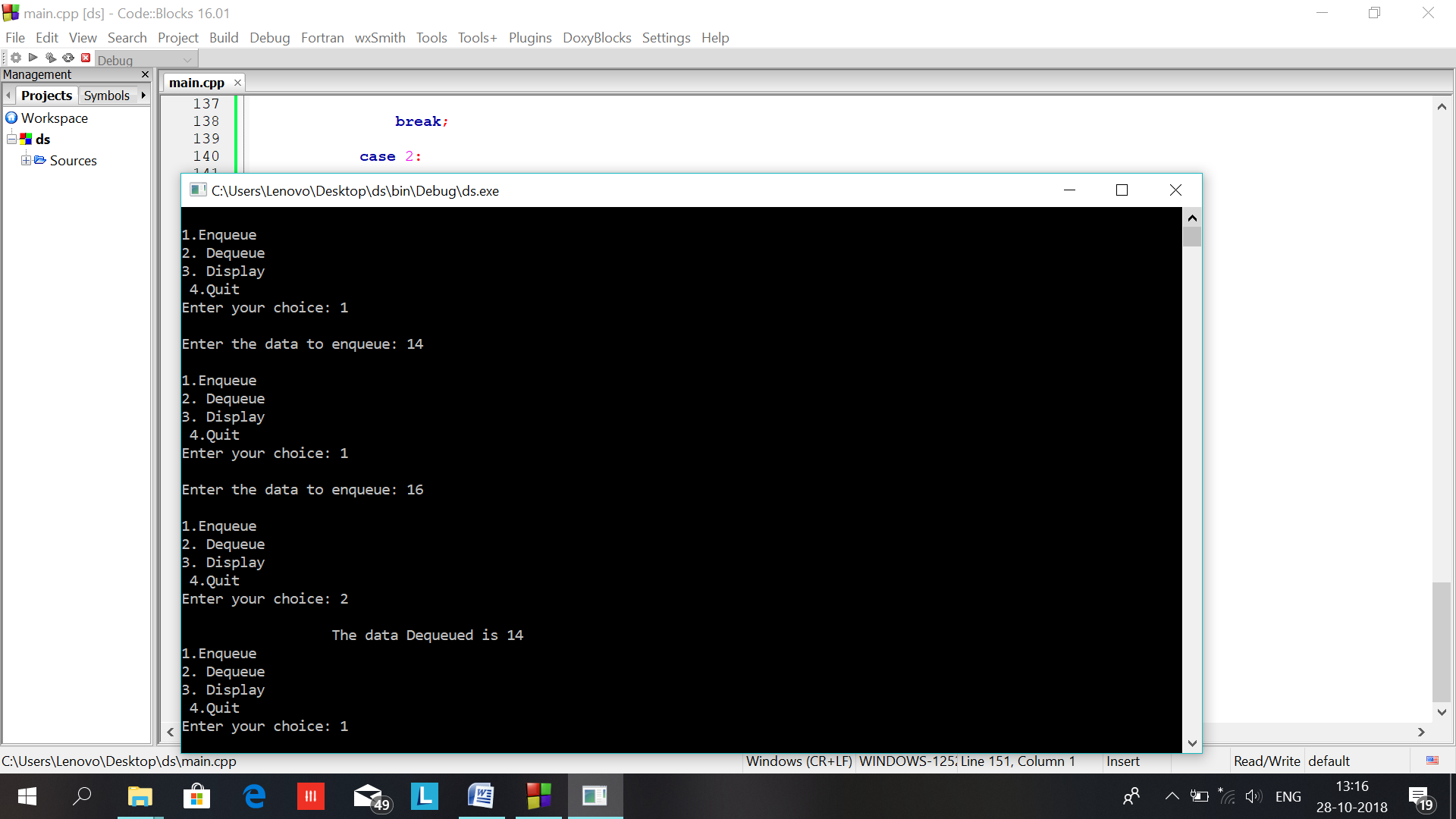
break;

}

}

return 0;

}



Ques:10

Ans:

#include<iostream>

using namespace std;

class node

{

public:

int power;

int coeff;

node \*next;

};

class poly

{

node \*first;

int n;

public:

poly()

{

first=NULL;

}

void create()

{

char ch;

first=new node;

cout<<"Enter power and coeff for first node respectively:\n";

cin>>first->power;

cin>>first->coeff;

first->next=NULL;

do

{

cout<<"Want to enter more data:\n";

cin>>ch;

if(ch=='y')

{

insert();

}

}while(ch=='y');

}

void insert()

{

node \*current,\*temp,\*prev;

temp=new node;

cout<<"Enter power and coeff respectively:\n";

cin>>temp->power;

cin>>temp->coeff;

int b=temp->power;

temp->next=NULL;

prev=NULL;

current=first;

while((current!=NULL)&&((current->power)<b))

{

prev=current;

current=current->next;

}

if(prev==NULL)

{

temp->next=first;

first=temp;

}

else if(current==NULL)

{

prev->next=temp;

//temp->next=NULL;

}

else

{

temp->next=current;

prev->next=temp;

}

}

void add(poly p1,poly p2)

{

cout<<"doign";

//poly p3;

node \*curr,\*curr1,\*temp,\*curr2;

curr=p1.first;

curr1=p2.first;

//curr2=p3.first;

while((curr!=NULL)&&(curr1!=NULL))

{

if(curr->power==curr1->power)

{

temp=new node;

temp->power=curr->power;

temp->coeff=curr->coeff+curr1->coeff;

temp->next=NULL;

curr=curr->next;

curr1=curr1->next;

}

else if(curr->power<curr1->power)

{

temp=new node;

temp->power=curr->power;

temp->coeff=curr->coeff;

temp->next=NULL;

curr=curr->next;

}

else if(curr1->power<curr->power)

{

temp=new node;

temp->power=curr1->power;

temp->coeff=curr1->coeff;

temp->next=NULL;

curr1=curr1->next;

}

cout<<"Niow linking starts:\n";

if(first==NULL)

{

first=temp;

curr2=first;

}

else

{

curr2->next=temp;

curr2=temp;

}

}

if(curr!=NULL)

{

while(curr!=NULL)

{

temp=new node;

temp->power=curr->power;

temp->coeff=curr->coeff;

temp->next=NULL;

curr=curr->next;

curr2->next=temp;

curr2=temp;

}

}

else if(curr1!=NULL)

{

while(curr1!=NULL)

{

temp=new node;

temp->power=curr1->power;

temp->coeff=curr1->coeff;

temp->next=NULL;

curr1=curr1->next;

curr2->next=temp;

curr2=temp;

}

}

if(curr2!=NULL)

cout<<"warning curr2 not null:\n";

}

void display()

{

node \*current;

current=this->first;

while(current!=NULL)

{

cout<<current->power<<" -> "<<current->coeff<<"\n";

current=current->next;

}

cout<<endl;

}

};

int main()

{

poly p1,p2,p3;

cout<<"Enter data of first polynomial:\n";

p1.create();

cout<<"Enter data for second polynomial:\n";

p2.create();

p1.display();

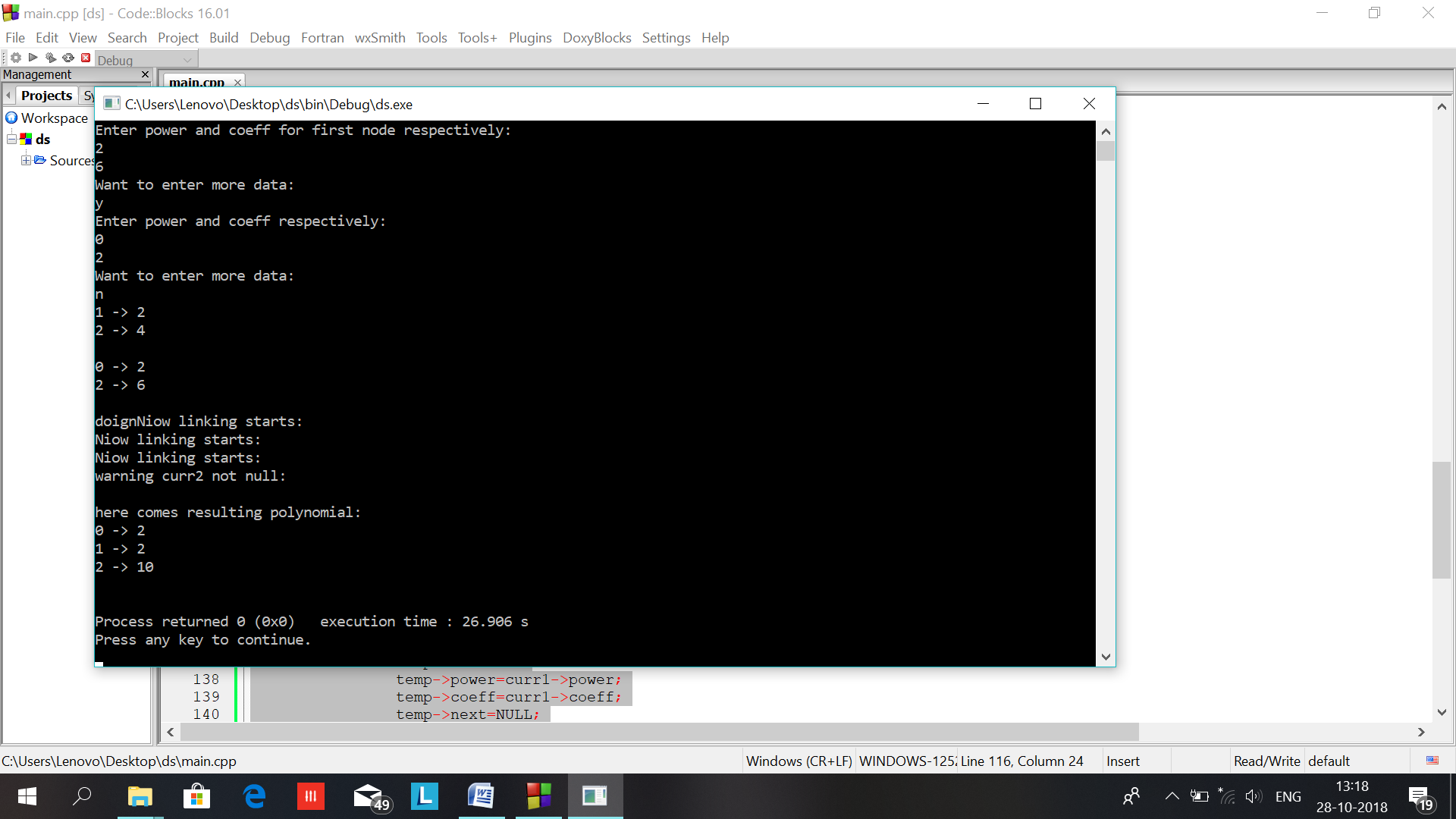
p2.display();

p3.add(p1,p2);

cout<<"\nhere comes resulting polynomial:\n";

p3.display();

}



Ques:11

Ans:

#include<iostream>

using namespace std;

void factor\_iter(int n) {

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

if(n%i==0)

cout<<i<<" ";

}

void factor\_rec(int n,int i=1) {

if(n<i)

return;

if(n%i==0)

cout<<i<<" ";

factor\_rec(n,++i);

}

long factorial\_iter(int n) {

long fact=1;

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

fact\*=i;

return fact;

}

long factorial\_rec(int n) {

if(n==1)

return 1;

return n\*factorial\_rec(n-1);

}

int main() {

cout<<"\nENTER A NUMBER : ";

int num;

cin>>num;

cout<<"\nFACTORS USING ITERATION : ";

factor\_iter(num);

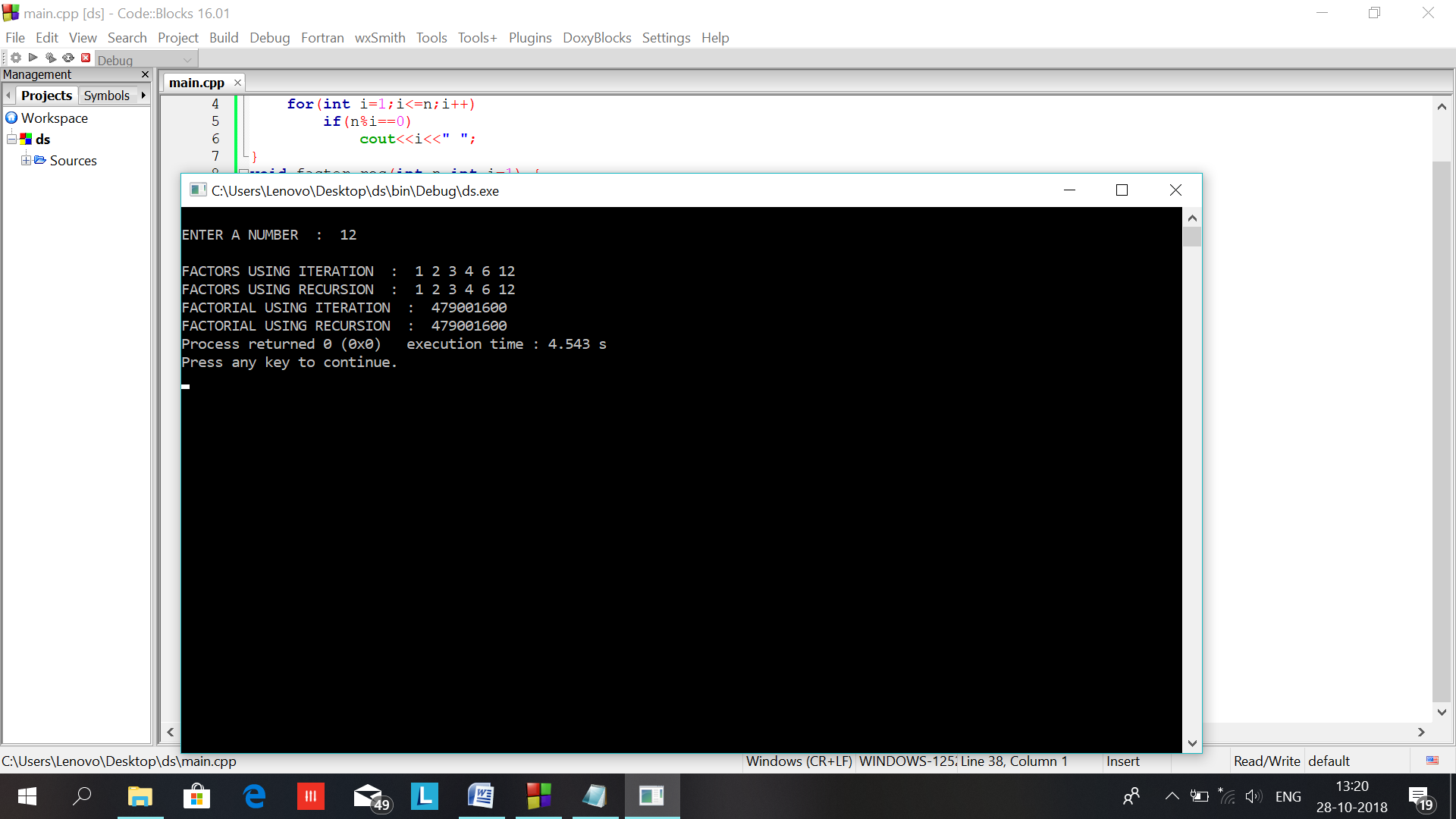
cout<<"\nFACTORS USING RECURSION : ";

factor\_rec(num);

cout<<"\nFACTORIAL USING ITERATION : "<<factorial\_iter(num);

cout<<"\nFACTORIAL USING RECURSION : "<<factorial\_rec(num);

}



Ques:12

Ans:

#include<iostream>

using namespace std;

void fib\_iter(int n) {

int first=0,second=1;

cout<<first<<" "<<second<<" ";

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

cout<<first+second<<" ";

int temp=first+second;

first=second;

second=temp;

}

}

void fib\_rec(int n,int first=0,int second=1) {

static int i=0;

if(i>=n)

return;

cout<<first<<" ";

i++;

fib\_rec(n,second,first+second);

}

int main() {

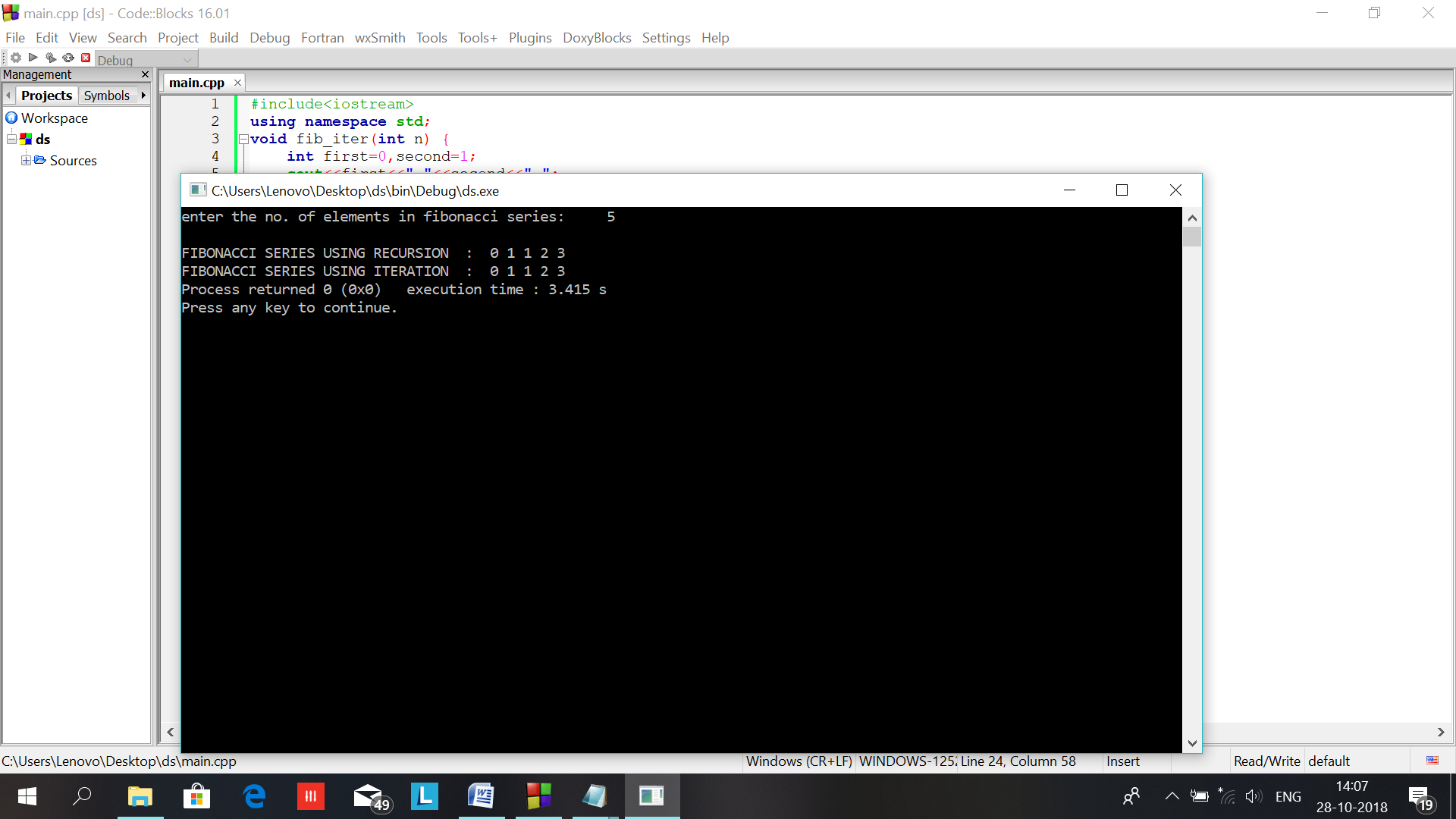
cout<<"\nFIBONACCI SERIES USING RECURSION : ";

fib\_rec(10);

cout<<"\nFIBONACCI SERIES USING ITERATION : ";

fib\_iter(10);

}



Ques:13

Ans:

#include<iostream>

using namespace std;

int GCD(int,int);

int main()

{

int a,b,rem,ch;

cout<<"\nEnter two numbers whose gcd is to be calculated: ";

cin>>a>>b;

cout<<"Enter your choice\n\n1.using recrsion\n2.without recursion\n";

cin>>ch;

if(ch==1)

cout<<"\nGCD of two numbers using recursion is: "<<GCD(a,b);

else if(ch==2)

{

cout<<"\nGCD of two numbers without using recursion is: ";

while(b%a!=0)

{

rem=b%a;

b=a;

a=rem;

}

cout<<a;

}

else

{

cout<<"enter correct choice\n";

}

return 0;

}

int GCD(int a,int b)

{

int rem;

if(b%a==0)

return a;

else

{

rem=b%a;

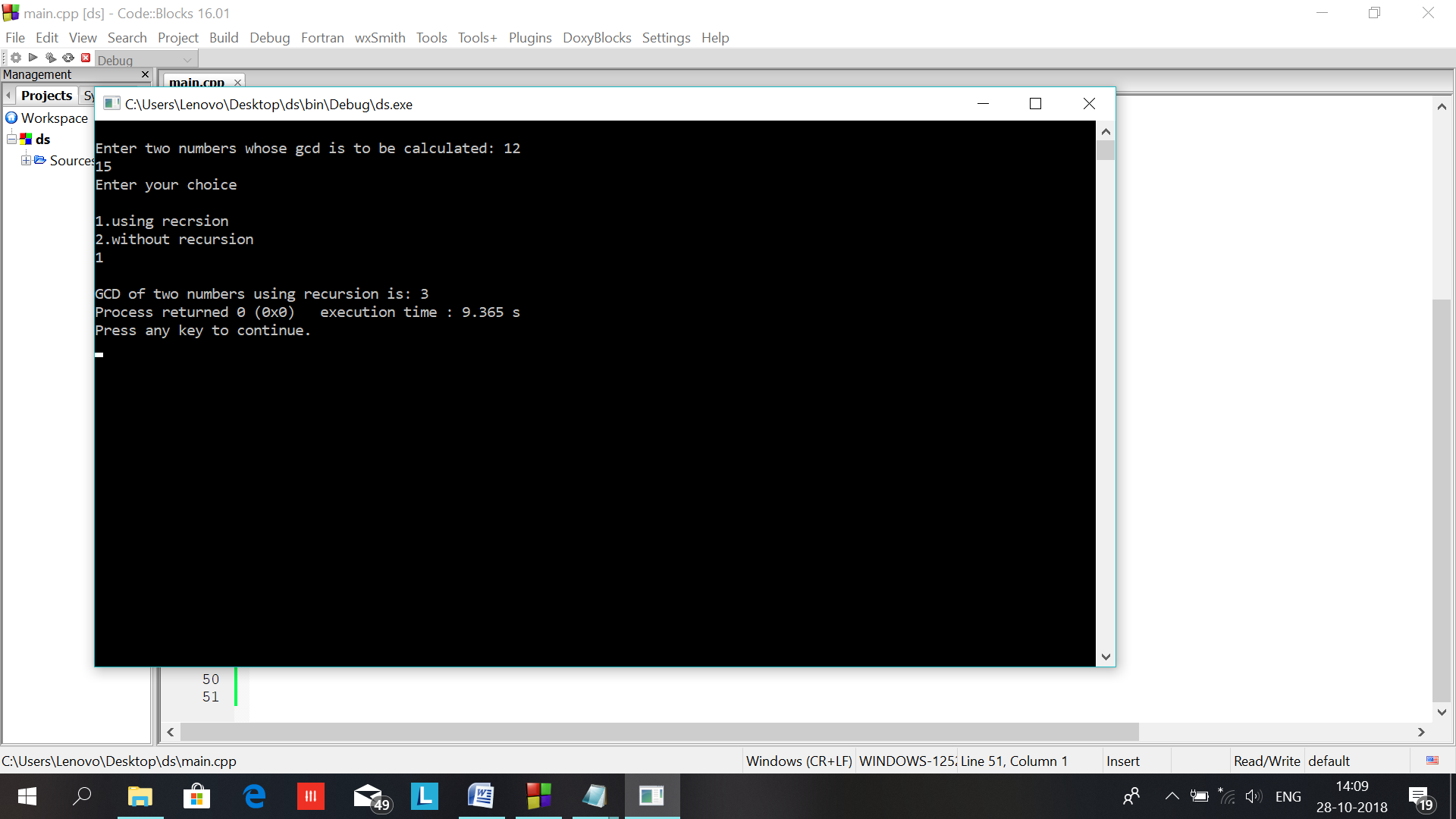
b=a;

a=rem;

GCD(a,b);

}

}



Ques:15

Ans:

#include<iostream>

using namespace std;

class node

{

public:

int a;

int b;

int c;

}obj[10];

void reterive( int m , int n)

{

int q=0;

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

{

cout<<endl;

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

{

if(obj[q].a==i&&obj[q].b==j)

{

cout<<obj[q].c<<" ";

q++;

}

else

cout<<"0 ";

}

}

};

int main()

{

int a[20][20];

int r,c;

int b[50];

cout<<"Enter the no. of rows in sparse matrix\n\n";

cin>>r;

cout<<"\nEnter the no. of columns in sparse matrix\n\n";

cin>>c;

cout<<"\nenter the elemnts in matrix"<<endl;

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

{

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

{

cin>>a[i][j];

}

}

cout<<"\n sparse matrix u entered is\n\n"<<endl;

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

{

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

{

cout<<" "<<a[i][j]<<" ";

}cout<<endl;

}

cout<<"\n\n corresponding non zero form for this is\n\n";

int k=0,d=0;

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

{

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

{

if(a[i][j]!=0)

{

obj[d].a=i;

obj[d].b=j;

obj[d].c=a[i][j];

d++;

b[k]=a[i][j];

k++;

}

}

}

int len=k;

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

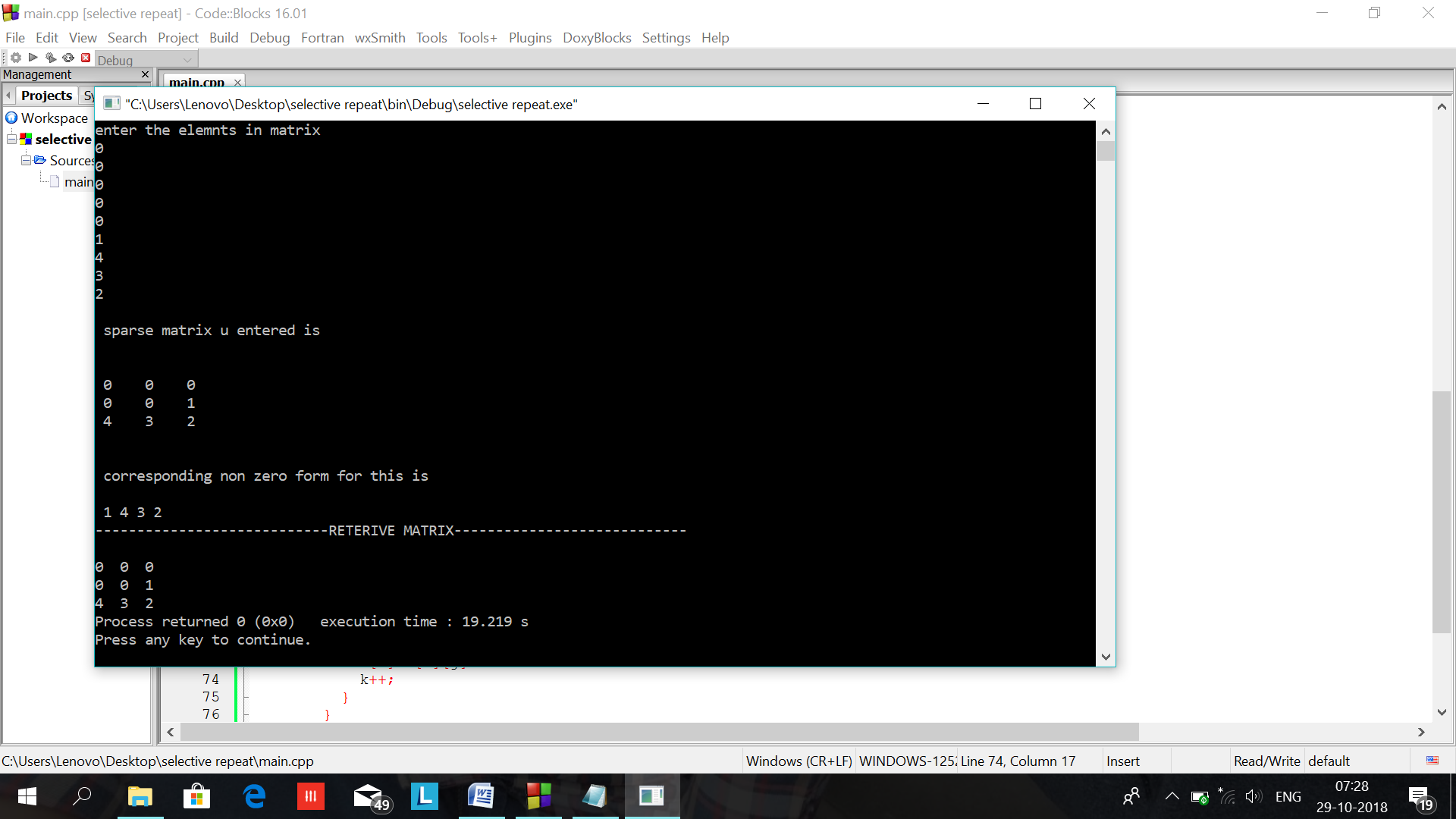
cout<<" "<<b[i];

cout<<"\n----------------------------RETERIVE MATRIX----------------------------\n";

reterive(r,c);

return 0;

}



Ques:16

Ans:

#include<iostream>

using namespace std;

#include<process.h>

class stack

{int data;

stack \*next,\*top;

public:

void push(int,int);

int pop();

void show();

void process(stack,int);

};

void stack::push(int info,int n)

{stack \*x;

x=new stack;

x->data=info;

if(n==0)

{

top=x;

top->next=NULL;

}else

{

x->next=top;

top=x;

}

}

int stack::pop()

{

int inf;

stack \*x;

x=top->next;

inf=top->data;

delete top;

top=x;

return inf;

}

void stack::show()

{

stack \*x;

int count;

x=top;

while(x!=NULL)

{

cout<<endl<<x->data;

x=x->next;

}

}

void stack::process(stack s,int n)

{

stack s2;

int info;

cout<<"\nOriginal stack is";

s.show();

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

{

info=s.pop();

s2.push(info,k);

}

cout<<"\nReversed stack is";

s2.show();

}

int main()

{

stack s,s1;

int info,ch,n;

cout<<"\nHow many elements do you want to insert\n";

cin>>n;

cout<<"\nEnter the elements to be inserted : \n";

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

{

cin>>info;

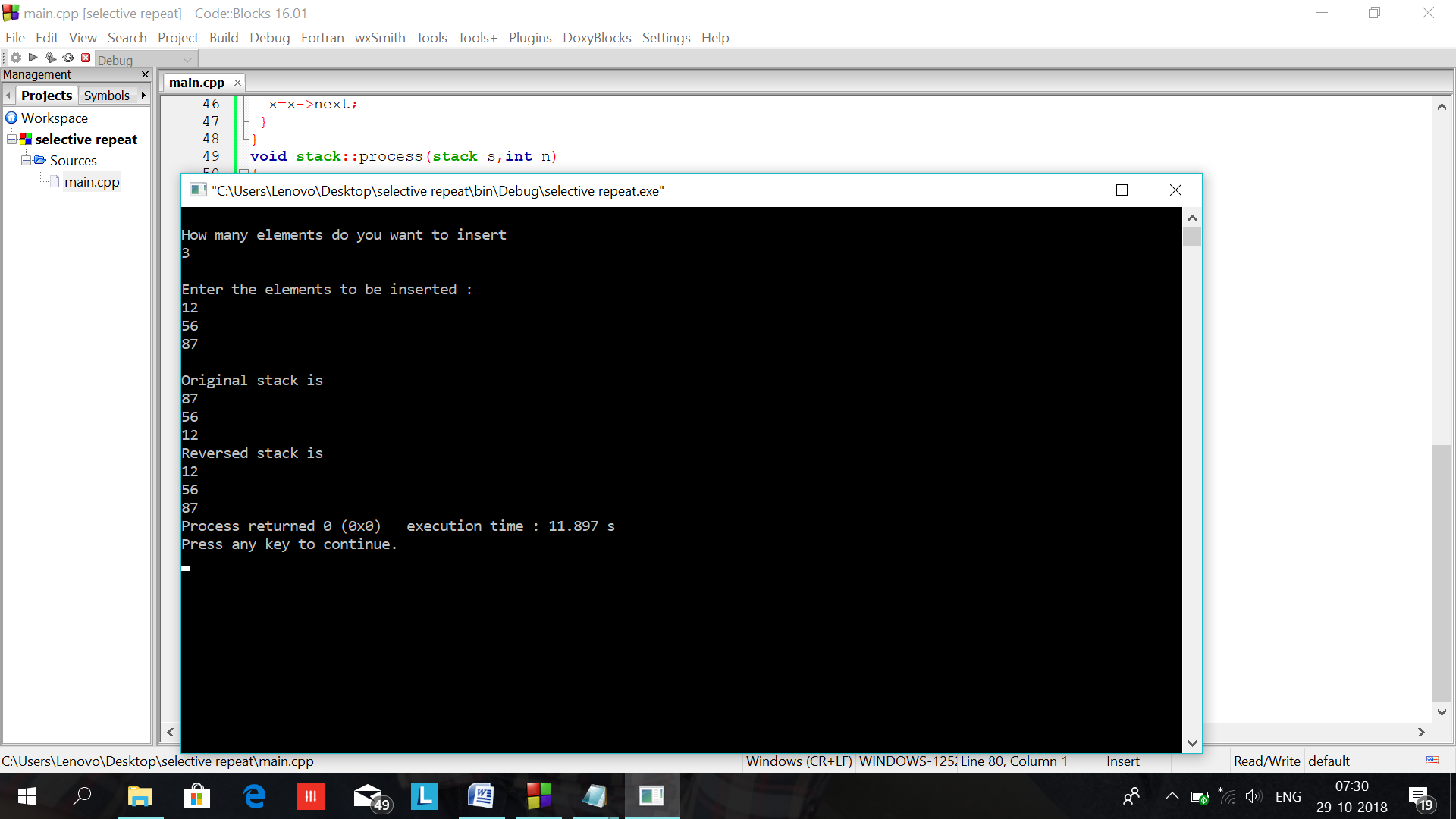
s.push(info,i);

}

s1.process(s,n);

return 0;

}



Ques:17

Ans:

#include<iostream>

using namespace std;

#define max 100

class queue

{

int front,rear;

int arr[max];

public:

queue()

{

front=rear=-1;

}

void add(int);

void display();

int del();

};

int queue::del()

{

int item;

item=arr[front];

front++;

return item;

}

void queue::add(int info)

{

rear++;

arr[rear]=info;

if(front==-1)front=0;

}

class stack

{

int top;

int stck[max];

public:stack()

{

top=-1;

}

void push(int item)

{

stck[++top]=item;

}

int pop()

{

return stck[top--];

}

void show()

{

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

cout<<stck[i]<<" ";

}

};

int main()

{

int n;

int item;

stack s;

queue a;

cout<<"\nHow many items you want to insert ";

cin>>n;

cout<<"\nEnter the elements you want to insert: ";

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

{

cin>>item;

s.push(item);

}

cout<<"\nOriginal stack: ";

s.show();

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

{

item=s.pop();

a.add(item);

}

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

{

item=a.del();

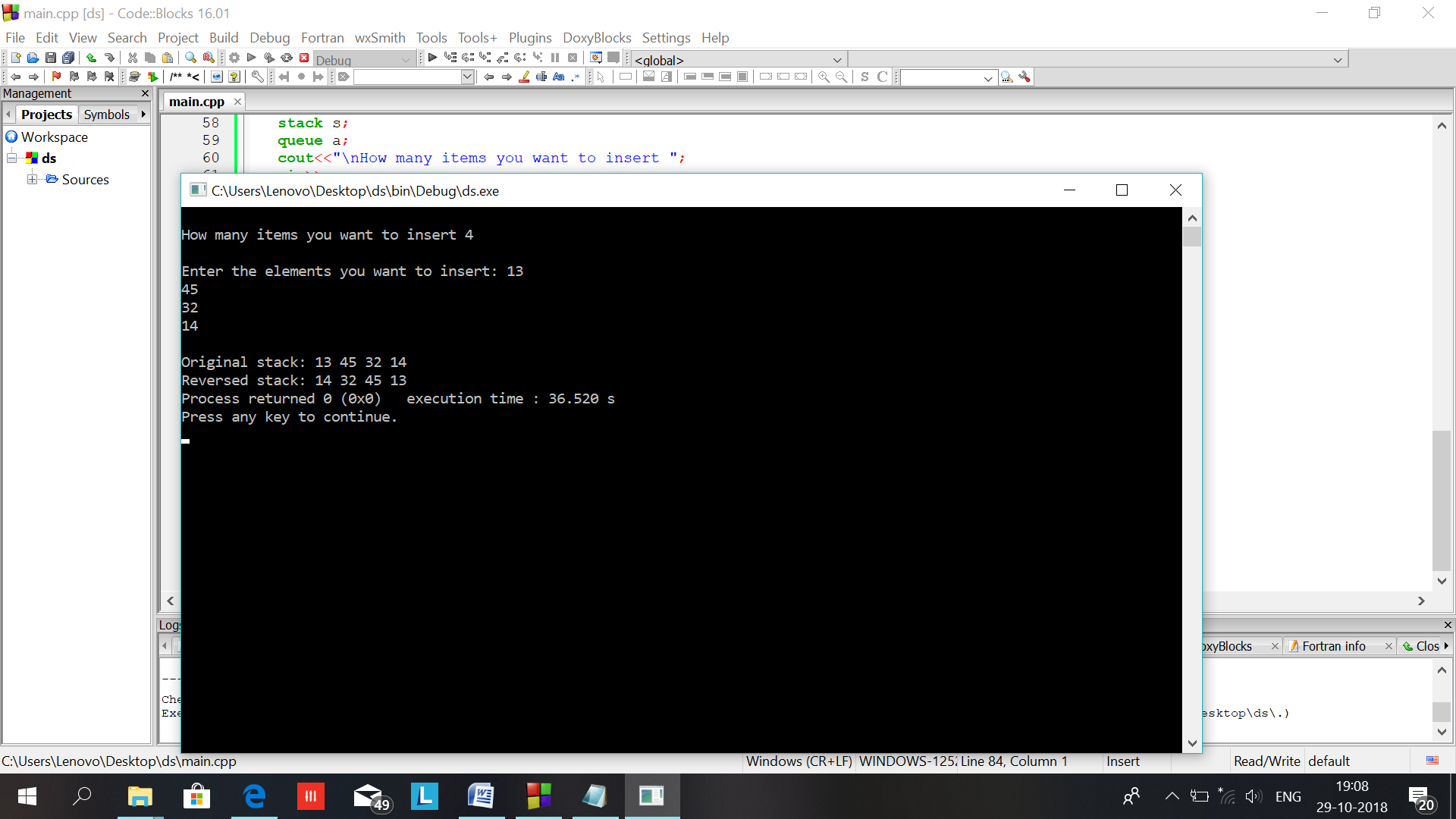
s.push(item);

}cout<<"\nReversed stack: ";

s.show();

return 0;

}



Ques:18

Ans:

#include<iostream>

using namespace std;

int main() {

int size;

cout<<"\nENTER SIZE OF ARRAY : ";

cin>>size;

int arr[size],matrix[size][size];

cout<<"\nENTER ELEMENTS IN ARRAY : ";

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

cin>>arr[i];

for(int i=0,k=0;i<size;i++)

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

if(i==j)

matrix[i][j]=arr[k++];

else

matrix[i][j]=0;

cout<<"\nDIAGONAL MATRIX :\n";

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

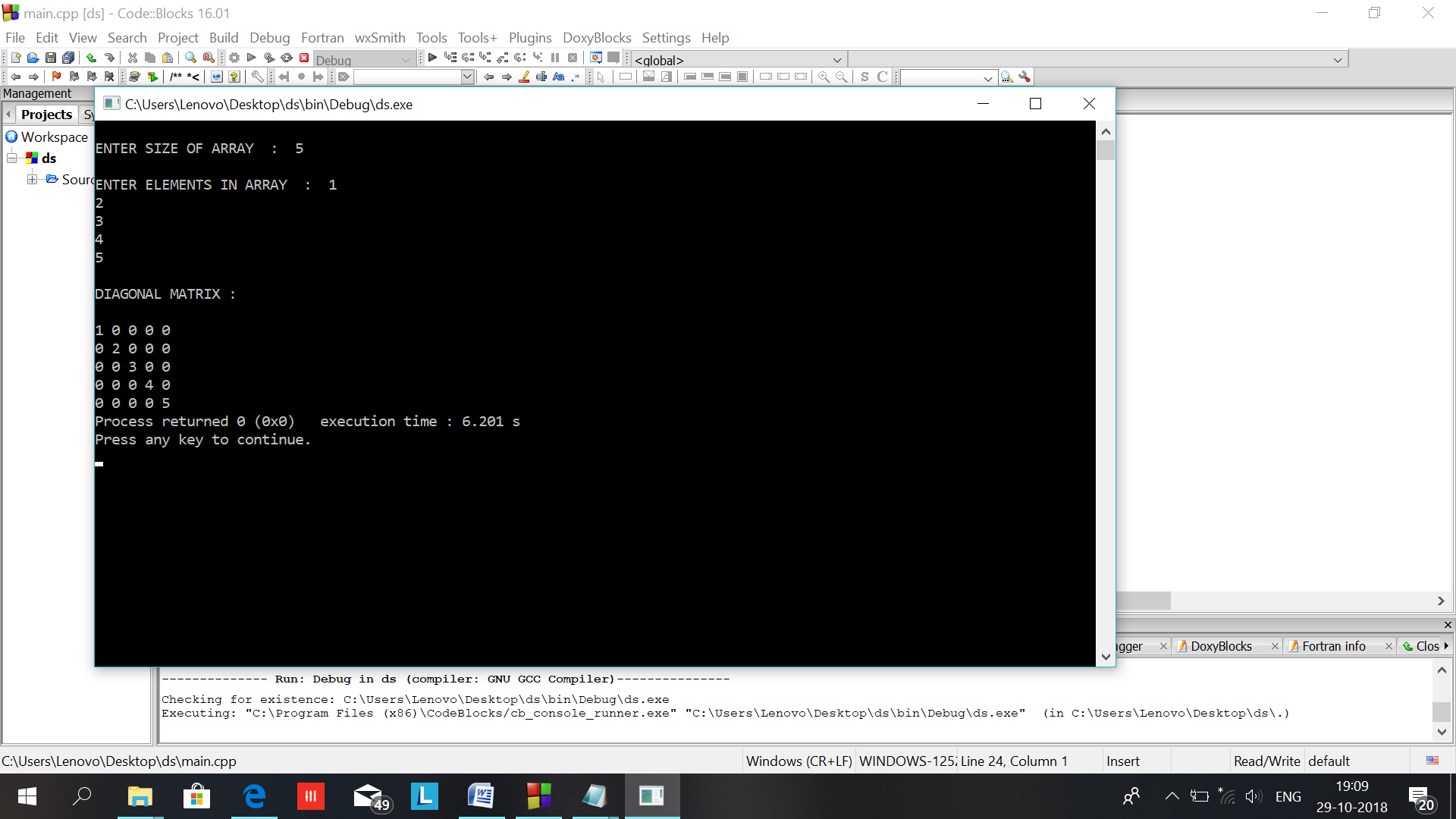
cout<<"\n";

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

cout<<matrix[i][j]<<" ";

}

}



Ques:19

Ans:

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int size1,a,b[15],c;

cout<<"------------LOWER TRIANGULAR MATRIX------------------------------\n";

cout<<"Enter the size of matrix : ";

cin>>size1;

int x=size1;

cout<<"\n Enter the row major of the matrix : ";

c=int(size1\*(size1+1)/2);

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

{

cin>>b[i];

}

cout<<"\n -----------------------------MATRIX----------------------------\n";

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

{

cout<<endl;

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

{

if(i>=j)

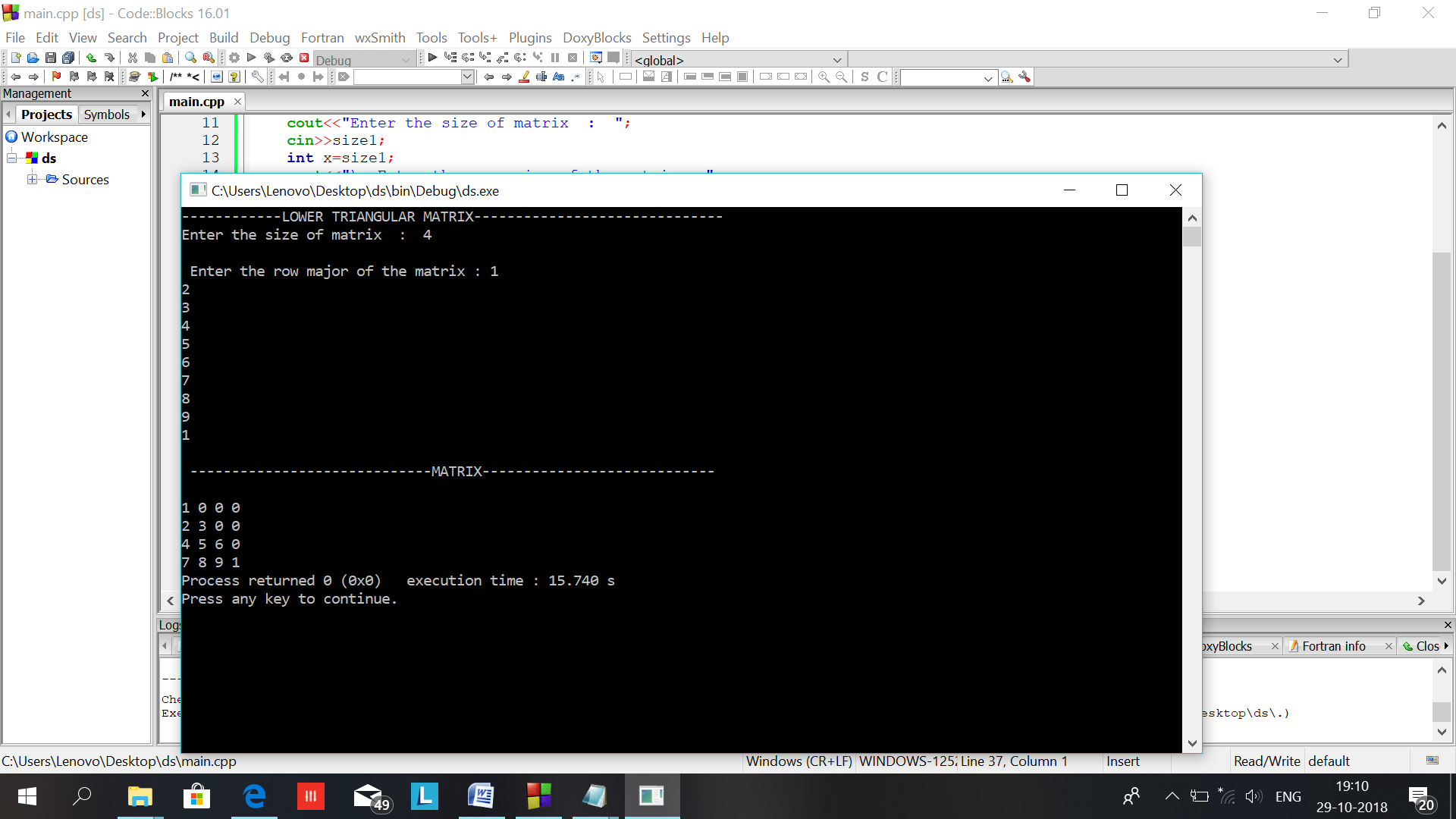
cout<<b[(i\*(i+1)/2)+j]<<" ";

else

cout<<"0 ";

}

}

}

Ques:20

Ans:

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int size1,a,b[15],c;

cout<<"------------UPPER TRIANGULAR MATRIX------------------------------\n";

cout<<"Enter the size of matrix : ";

cin>>size1;

int x=size1;

cout<<"\n Enter the column major of the matrix : ";

c=int(size1\*(size1+1)/2);

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

{

cin>>b[i];

}

cout<<"\n -----------------------------MATRIX----------------------------\n";

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

{

cout<<endl;

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

{

if(i<=j)

cout<<b[((j\*(j-1)/2)+i)]<<" ";

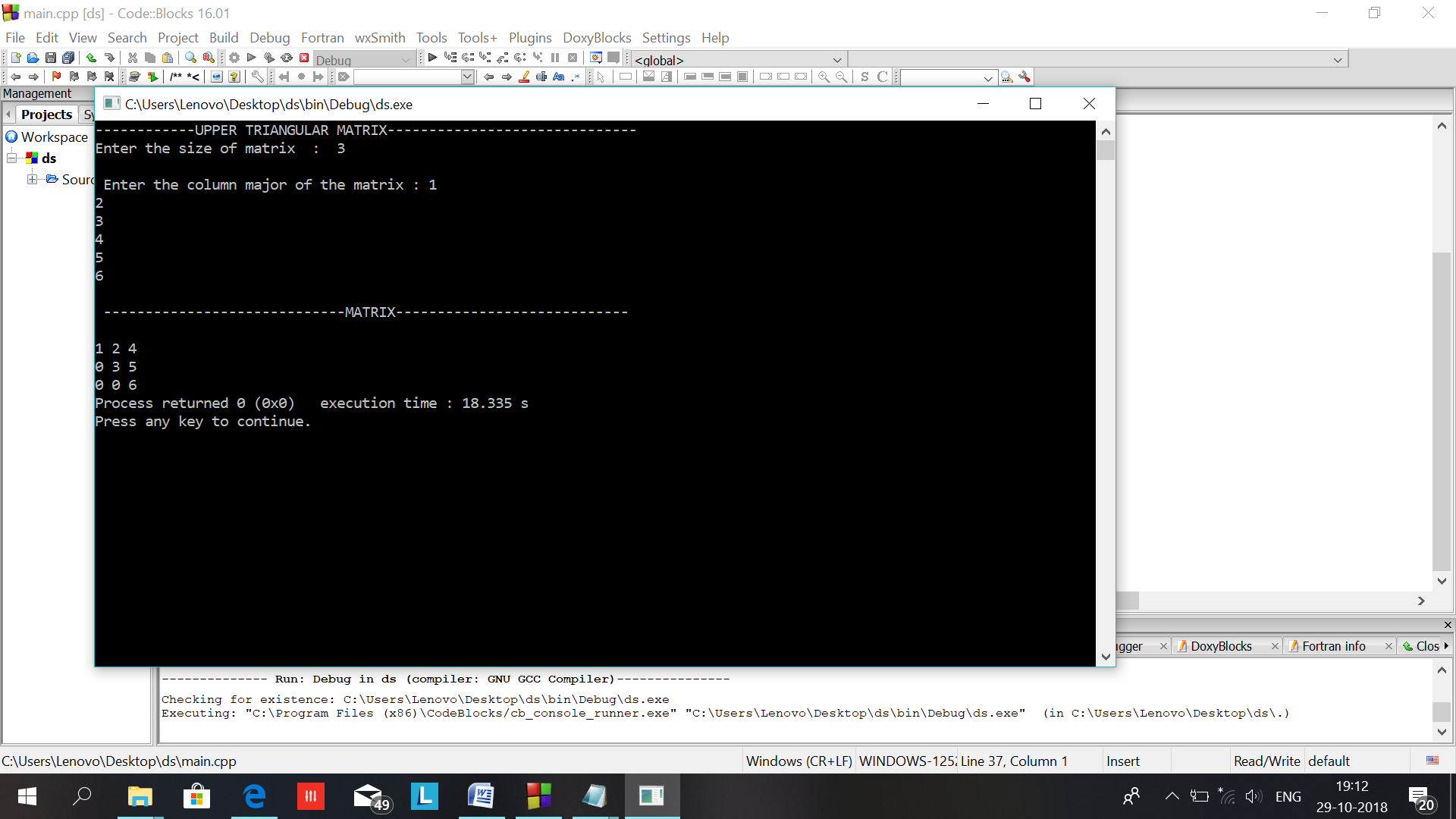
else

cout<<"0 ";

}

}

}



Ques:21

Ans:

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int size1,a,b[15],c;

cout<<"------------SYMMETRIX TRIANGULAR MATRIX------------------------------\n";

cout<<"Enter the size of matrix : ";

cin>>size1;

int x=size1;

cout<<"\n Enter the row major of the matrix : ";

c=int(size1\*(size1+1)/2);

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

{

cin>>b[i];

}

cout<<"\n -----------------------------MATRIX----------------------------\n";

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

{

cout<<endl;

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

{

if(i>=j)

cout<<b[(i\*(i+1)/2)+j]<<" ";

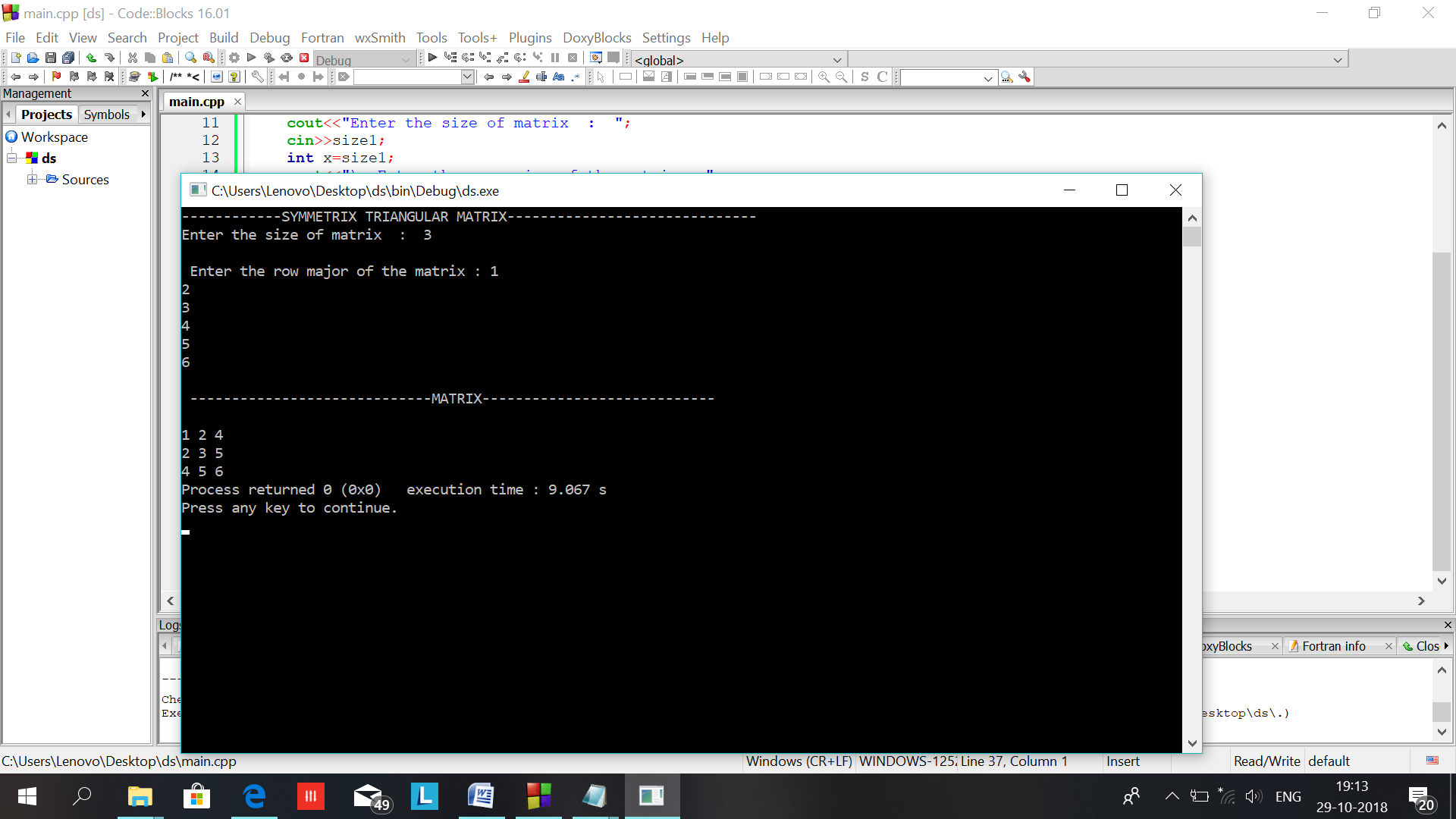
else

cout<<b[(j\*(j+1)/2)+i]<<" ";

}

}

}



Ques:22

Ans:

#include<iostream>

#include <stdio.h>

#include <conio.h>

#include <stdlib.h>

using namespace std;

typedef struct threadedbinarytree

{

int info;

struct threadedbinarytree \*left,\*right;

int lflag,rflag;

}tbt;

tbt \*create();

tbt \*makenode(int);

tbt \*insert(tbt \*,tbt \*);

int inorderSuccessor(tbt \*,tbt \*);

int inorderPredecessor(tbt \*,tbt \*);

void traverseInorder(tbt \*);

/\*Allocate memory\*/

tbt \*create()

{

tbt \*temp=(tbt \*)malloc(sizeof(tbt));

if(temp==NULL)

{cout<<"\nMemory Allocation Error!...";

exit(1);

}

return temp;

}

/\*Make a new node\*/

tbt \*makenode(int x)

{

tbt \*temp=create();

temp->info=x;

temp->left=NULL;

temp->right=NULL;

temp->lflag=1; /\*default is thread\*/

temp->rflag=1;/\*default is thread\*/

return temp;

}

/\*Insert a newnode in the threaded tree\*/

tbt \*insert(tbt \*root,tbt \*newnode)

{

tbt \*p,\*temp=root;

if(root==NULL)

{root=newnode;

return root;

}

while(temp!=NULL)

{

if(newnode->info<temp->info)

{

if(temp->left)

{if(temp->lflag==1)

{

newnode->left=temp->left;

newnode->right=temp;

temp->left=newnode;

temp->lflag=0;

break;

}

else

temp=temp->left;

}

else

{newnode->left=temp->left;

temp->left=newnode;

temp->lflag=0;

newnode->right=temp;

break;

}

}

else

{if(temp->right)

{if(temp->rflag==1)

{newnode->right=temp->right;

newnode->left=temp;

temp->right=newnode;

temp->rflag=0;

newnode->rflag=1;

break;

}

else

{temp=temp->right;

}

}

else

{

p=temp->right;

temp->right=newnode;

temp->rflag=0;

newnode->right=p;

newnode->left=temp;

break;

}

}

}

return root;

}

/\*Function that returns Inorder successor\*/

int inorderSuccessor(tbt \*root,tbt \*node)

{int num;

tbt \*p,\*temp=root;

while(temp!=NULL)

{

if(node->info<temp->info)

{if(temp->lflag==0)

temp=temp->left;

else

temp=temp->right;

}

else if(node->info>temp->info)

{

if(temp->rflag==0)

temp=temp->right;

else

{cout<<"\nGiven node does not exist...";

return -1;

}

}

else/\*node->info==temp->info\*/

{

if(temp->rflag==0)

{

p=temp->right;

while(p->lflag!=1)

p=p->left;

num=p->info;/\*successor\*/

break;

}

else

{

if(temp->right==NULL)

{num=0;

break;

}

else

{num=temp->right->info;

break;

}

}

}

}

return num;

}

/\*Function that returns inorder predecessor\*/

int inorderPredecessor(tbt \*root,tbt \*node)

{

int num;

tbt \*p,\*temp=root;

while(temp!=NULL)

{

if(node->info<temp->info)

{

if(temp->lflag==0)

temp=temp->left;

else /\*move from left to right\*/

temp=temp->right;

}

else if(node->info>temp->info)

{if(temp->rflag==0)

temp=temp->right;

else

{cout<<"\nGiven node does not exist....";

return -1;

}

}

else/\*if value is found\*/

{if(temp->lflag==0)

{p=temp->left;

while(p->rflag!=1)

p=p->right;

num=p->info;/\*predecssor\*/

break;

}

else

{if(temp->left==NULL)

{num=0;

break;

}

else

{num=temp->left->info;

break;

}

}

}

}

return num;

}

/\*Traverse in Inorder\*/

void traverseInorder(tbt \*root)

{tbt \*temp=root;

while(temp->left)

temp=temp->left;

while(temp!=NULL)

{cout<<" "<<temp->info;

if(temp->rflag==0)

{temp=temp->right;

while(temp->lflag==0)

temp=temp->left;

}

else

{temp=temp->right;

}

}

}

int main()

{

int x,ch,num;

tbt \*node, \*root=NULL;

while(1)

{

//clrscr();

cout<<"\nMenu-In Threaded Binary Tree";

cout<<"\n 1. Insert";

cout<<"\n 2. Inorder Traversal";

cout<<"\n 3. Return Inorder Predecessor";

cout<<"\n 4. Return Inorder Successor";

cout<<"\n 5. Exit";

cout<<"\nEnter your choice(1 to 5)";

cin>>ch;

switch(ch)

{

case 1:

cout<<"/nEnter the number...";

cin>>x;

node=makenode(x);

root=insert(root,node);

break;

case 2:

traverseInorder(root);

break;

case 3:

cout<<"\nEnter the node whose predecessor is required?";

cin>>num;

node=makenode(num);

x=inorderPredecessor(root,node);

if(x!=-1)

cout<<"\nInorder predecessor of "<<node->info<<" is "<<x;

else

cout<<"\nWrong node entered....";

break;

case 4:

cout<<"\nEnter the node whose successor is required?";

cin>>num;

node=makenode(num);

x=inorderSuccessor(root,node);

if(x!=-1)

cout<<"\nInorder successor of"<<node->info<<"is"<<x;

else

cout<<"\nWrong node entered...";

break;

case 5:

exit(0);

default:

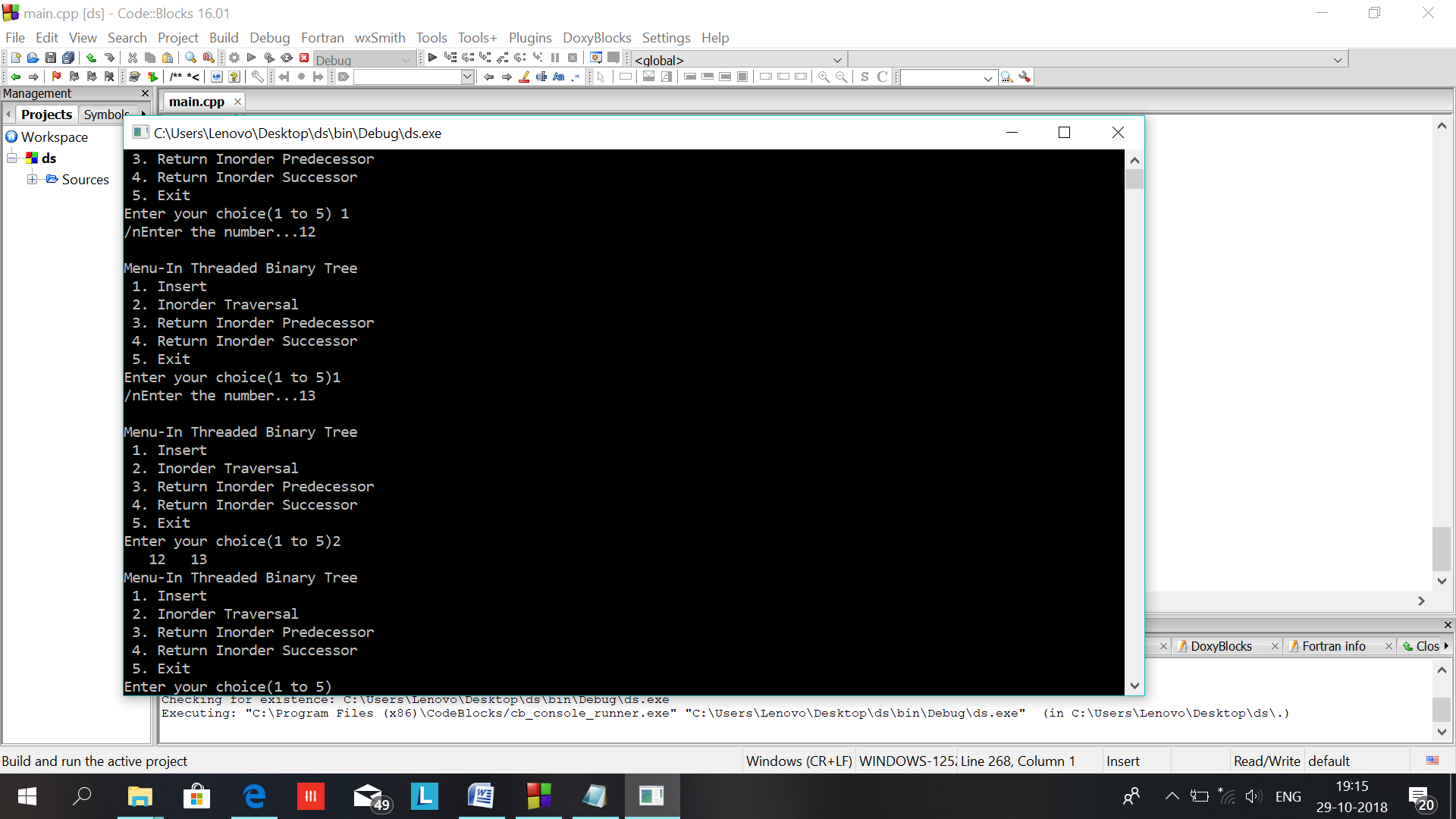
cout<<"\nInvalid Choice…!";

}

//getch();

}

}



Ques:23

Ans:

#include<iostream>

#include<cstdio>

#include<sstream>

#include<algorithm>

#define pow2(n) (1 << (n))

using namespace std;

/\*

\* Node Declaration

\*/

struct avl\_node

{

int data;

struct avl\_node \*left;

struct avl\_node \*right;

}\*root;

/\*

\* Class Declaration

\*/

class avlTree

{

public:

int height(avl\_node \*);

int diff(avl\_node \*);

avl\_node \*rr\_rotation(avl\_node \*);

avl\_node \*ll\_rotation(avl\_node \*);

avl\_node \*lr\_rotation(avl\_node \*);

avl\_node \*rl\_rotation(avl\_node \*);

avl\_node\* balance(avl\_node \*);

avl\_node\* insert(avl\_node \*, int );

void display(avl\_node \*, int);

void inorder(avl\_node \*);

void preorder(avl\_node \*);

void postorder(avl\_node \*);

avlTree()

{

root = NULL;

}

};

/\*

\* Main Contains Menu

\*/

int main()

{

int choice, item;

avlTree avl;

while (1)

{

cout<<"\n---------------------"<<endl;

cout<<"AVL Tree Implementation"<<endl;

cout<<"\n---------------------"<<endl;

cout<<"1.Insert Element into the tree"<<endl;

cout<<"2.Display Balanced AVL Tree"<<endl;

cout<<"3.InOrder traversal"<<endl;

cout<<"4.PreOrder traversal"<<endl;

cout<<"5.PostOrder traversal"<<endl;

cout<<"6.Exit"<<endl;

cout<<"Enter your Choice: ";

cin>>choice;

switch(choice)

{

case 1:

cout<<"Enter value to be inserted: ";

cin>>item;

root = avl.insert(root, item);

break;

case 2:

if (root == NULL)

{

cout<<"Tree is Empty"<<endl;

continue;

}

cout<<"Balanced AVL Tree:"<<endl;

avl.display(root, 1);

break;

case 3:

cout<<"Inorder Traversal:"<<endl;

avl.inorder(root);

cout<<endl;

break;

case 4:

cout<<"Preorder Traversal:"<<endl;

avl.preorder(root);

cout<<endl;

break;

case 5:

cout<<"Postorder Traversal:"<<endl;

avl.postorder(root);

cout<<endl;

break;

case 6:

exit(1);

break;

default:

cout<<"Wrong Choice"<<endl;

}

}

return 0;

}

/\*

\* Height of AVL Tree

\*/

int avlTree::height(avl\_node \*temp)

{

int h = 0;

if (temp != NULL)

{

int l\_height = height (temp->left);

int r\_height = height (temp->right);

int max\_height = max (l\_height, r\_height);

h = max\_height + 1;

}

return h;

}

/\*

\* Height Difference

\*/

int avlTree::diff(avl\_node \*temp)

{

int l\_height = height (temp->left);

int r\_height = height (temp->right);

int b\_factor= l\_height - r\_height;

return b\_factor;

}

/\*

\* Right- Right Rotation

\*/

avl\_node \*avlTree::rr\_rotation(avl\_node \*parent)

{

avl\_node \*temp;

temp = parent->right;

parent->right = temp->left;

temp->left = parent;

return temp;

}

/\*

\* Left- Left Rotation

\*/

avl\_node \*avlTree::ll\_rotation(avl\_node \*parent)

{

avl\_node \*temp;

temp = parent->left;

parent->left = temp->right;

temp->right = parent;

return temp;

}

/\*

\* Left - Right Rotation

\*/

avl\_node \*avlTree::lr\_rotation(avl\_node \*parent)

{

avl\_node \*temp;

temp = parent->left;

parent->left = rr\_rotation (temp);

return ll\_rotation (parent);

}

/\*

\* Right- Left Rotation

\*/

avl\_node \*avlTree::rl\_rotation(avl\_node \*parent)

{

avl\_node \*temp;

temp = parent->right;

parent->right = ll\_rotation (temp);

return rr\_rotation (parent);

}

/\*

\* Balancing AVL Tree

\*/

avl\_node \*avlTree::balance(avl\_node \*temp)

{

int bal\_factor = diff (temp);

if (bal\_factor > 1)

{

if (diff (temp->left) > 0)

temp = ll\_rotation (temp);

else

temp = lr\_rotation (temp);

}

else if (bal\_factor < -1)

{

if (diff (temp->right) > 0)

temp = rl\_rotation (temp);

else

temp = rr\_rotation (temp);

}

return temp;

}

/\*

\* Insert Element into the tree

\*/

avl\_node \*avlTree::insert(avl\_node \*root, int value)

{

if (root == NULL)

{

root = new avl\_node;

root->data = value;

root->left = NULL;

root->right = NULL;

return root;

}

else if (value < root->data)

{

root->left = insert(root->left, value);

root = balance (root);

}

else if (value >= root->data)

{

root->right = insert(root->right, value);

root = balance (root);

}

return root;

}

/\*

\* Display AVL Tree

\*/

void avlTree::display(avl\_node \*ptr, int level)

{

int i;

if (ptr!=NULL)

{

display(ptr->right, level + 1);

printf("\n");

if (ptr == root)

cout<<"Root -> ";

for (i = 0; i < level && ptr != root; i++)

cout<<" ";

cout<<ptr->data;

display(ptr->left, level + 1);

}

}

/\*

\* Inorder Traversal of AVL Tree

\*/

void avlTree::inorder(avl\_node \*tree)

{

if (tree == NULL)

return;

inorder (tree->left);

cout<<tree->data<<" ";

inorder (tree->right);

}

/\*

\* Preorder Traversal of AVL Tree

\*/

void avlTree::preorder(avl\_node \*tree)

{

if (tree == NULL)

return;

cout<<tree->data<<" ";

preorder (tree->left);

preorder (tree->right);

}

/\*

\* Postorder Traversal of AVL Tree

\*/

void avlTree::postorder(avl\_node \*tree)

{

if (tree == NULL)

return;

postorder ( tree ->left );

postorder ( tree ->right );

cout<<tree->data<<" ";

}

