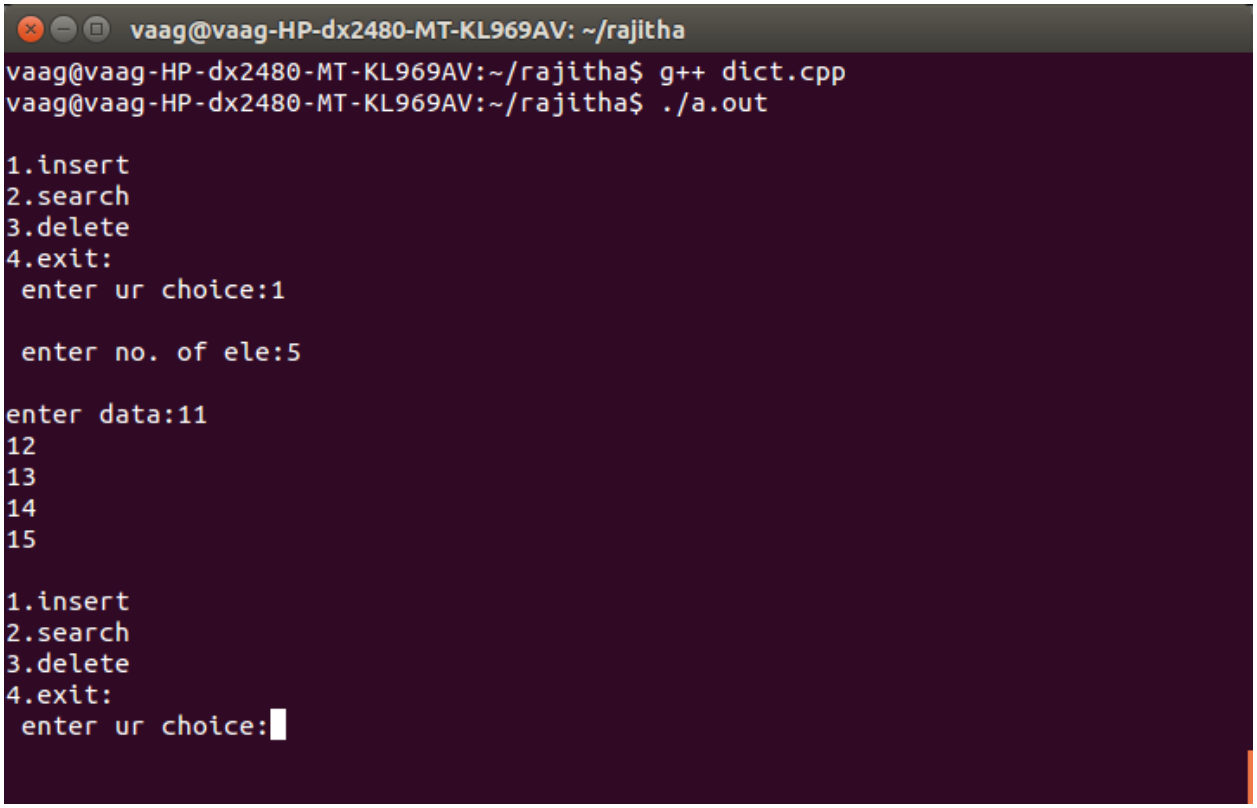
1. Write a C++ program to implement all the functions of a dictionary ADT.

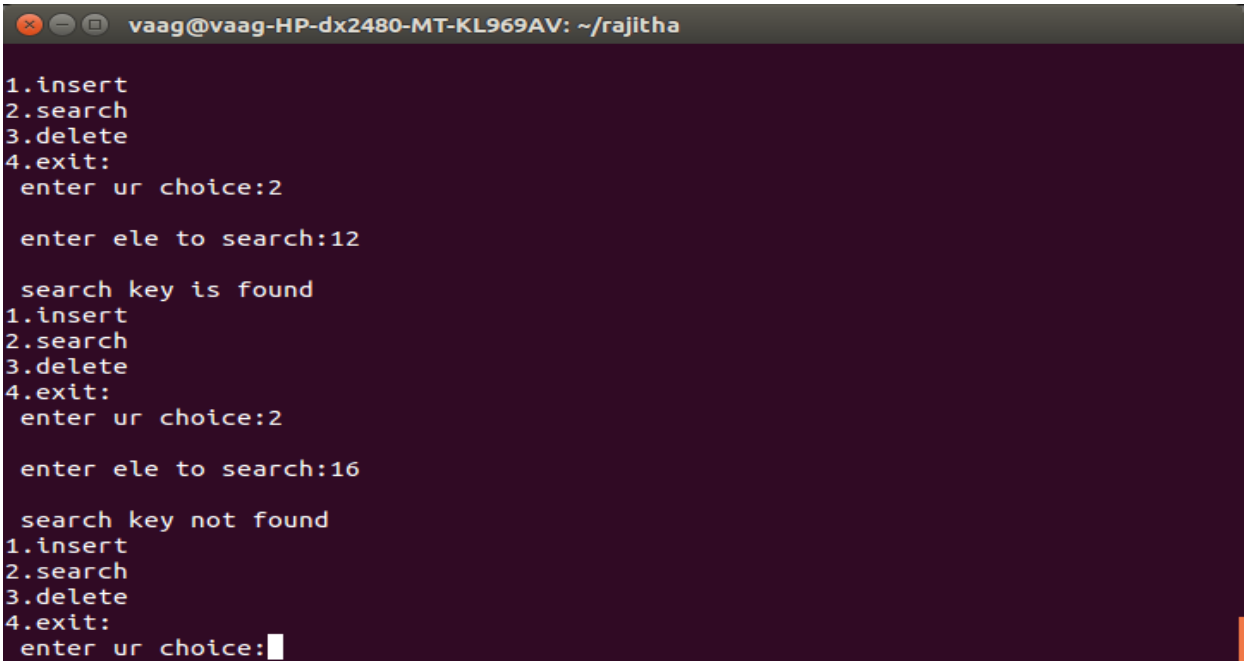
```
#include<iostream>
#include<stdlib.h>
using namespace std;
#define max 10
class dic
{
    public:

    struct list
    {
            int data;
            struct list *next;
    }*ptr[max],*root[max],*temp[max];
    int index;
    dic()
    {
            int i;
            index=-1;
            for(i=0;i<max;i++)
            {
                    root[i]=NULL;
                    ptr[i]=NULL;
                    temp[i]=NULL;
            }
    }
    void insert(int );
    void search(int );
    void dele(int);
    void disp();
};
void dic::insert(int key)
{
    index=int(key%max);
    ptr[index]=(struct list*)malloc(sizeof(struct list));
    ptr[index]->data=key;
    if(root[index]==NULL)
    {
            root[index]=ptr[index];
            root[index]->next=NULL;
            temp[index]=ptr[index];
    }
    else
    {
            temp[index]=root[index];
            while(temp[index]->next!=NULL)
                    temp[index]=temp[index]->next;
            temp[index]->next=ptr[index];
    }
```

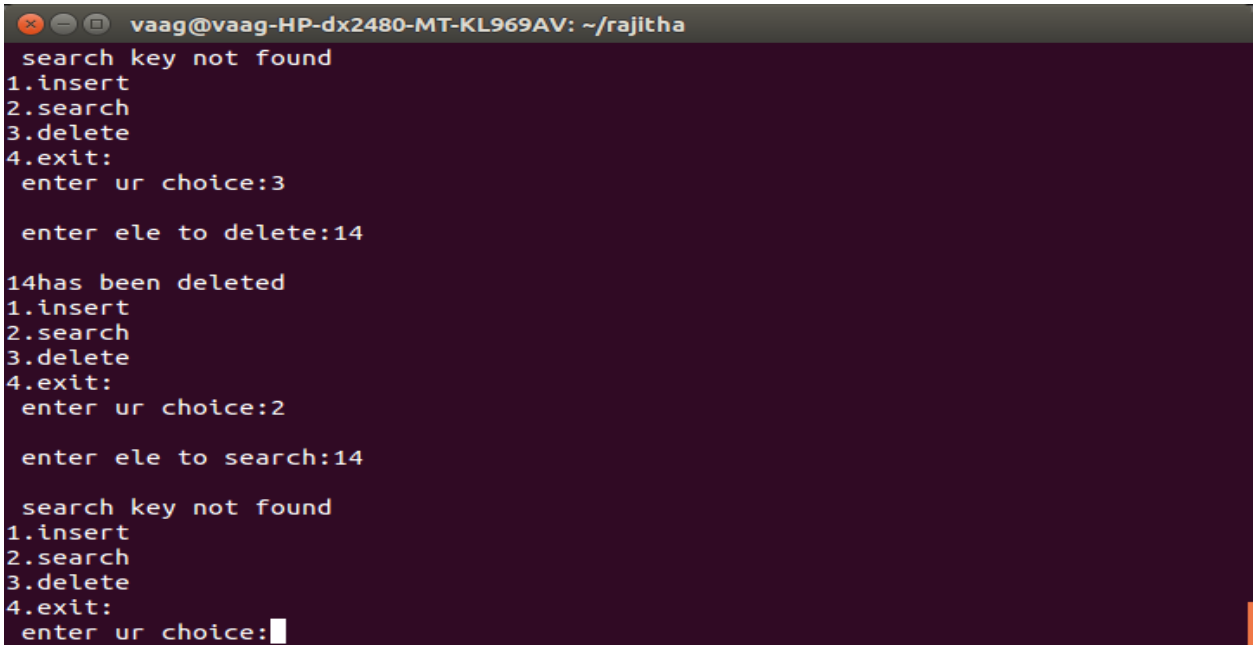
```
}
void dic::search(int key)
{
    int flag=0;
    index=int(key%max);
    temp[index]=root[index];
    while(temp[index]!=NULL)
    {
        if(temp[index]->data==key)
        {
            cout<<"\n search key is found";
            flag=1;
            break;
        }
        else
            temp[index]=temp[index]->next;
    }
    if(flag==0)
        cout<<"\n search key not found";
}
void dic::dele(int key)
{
    index=int(key%max);
    temp[index]=root[index];
    while(temp[index]->data!=key && temp[index]!=NULL)
    {
        ptr[index]=temp[index];
        temp[index]=temp[index]->next;
    }
    ptr[index]->next=temp[index]->next;
    cout<<"\n"<<temp[index]->data<<"has been deleted";
    temp[index]->data=0;
    temp[index]=NULL;
    free(temp[index]);
}
int main()
{
    int val,ch,n,num;
    char c;
    dic d;
    while(1)
    {
        cout<<"\n1.insert\n2.search\n3.delete\n4.exit:";
        cout<<"\n enter ur choice:";
        cin>>ch;
        switch(ch)
        {
            case 1:cout<<"\n enter no. of ele:";
                cin>>n;
                cout<<"\nenter data:";
                for(int i=0;i<n;i++)
```

```
                        {
                                cin>>num;
                                d.insert(num);
                         }
                         break;
                case 2:cout<<"\n enter ele to search:";
                         cin>>n;
                         d.search(n);
                         break;
                case 3:cout<<"\n enter ele to delete:";
                        cin>>n;
                        d.dele(n);
                        break;
                case 4:exit(0);
        }
    }
}
```

output:




vaag@vaag-HP-dx2480-MT-KL969AV: ~/rajitha
1.insert
2.search
3.delete
4.exit:
enter ur choice:2
enter ele to search:12
search key is found
1.insert
2.search
3.delete
4.exit:
enter ur choice:2
enter ele to search:16
search key not found
1.insert
2.search
3.delete
4.exit:
enter ur choice:


vaag@vaag-HP-dx2480-MT-KL969AV: ~/rajitha
search key not found
1.insert
2.search
3.delete
4.exit:
enter ur choice:3
enter ele to delete:14
14has been deleted
1.insert
2.search
3.delete
4.exit:
enter ur choice:2
enter ele to search:14
search key not found
1.insert
2.search
3.delete
4.exit:
enter ur choice:

2. Write a C++ program for skip lists.

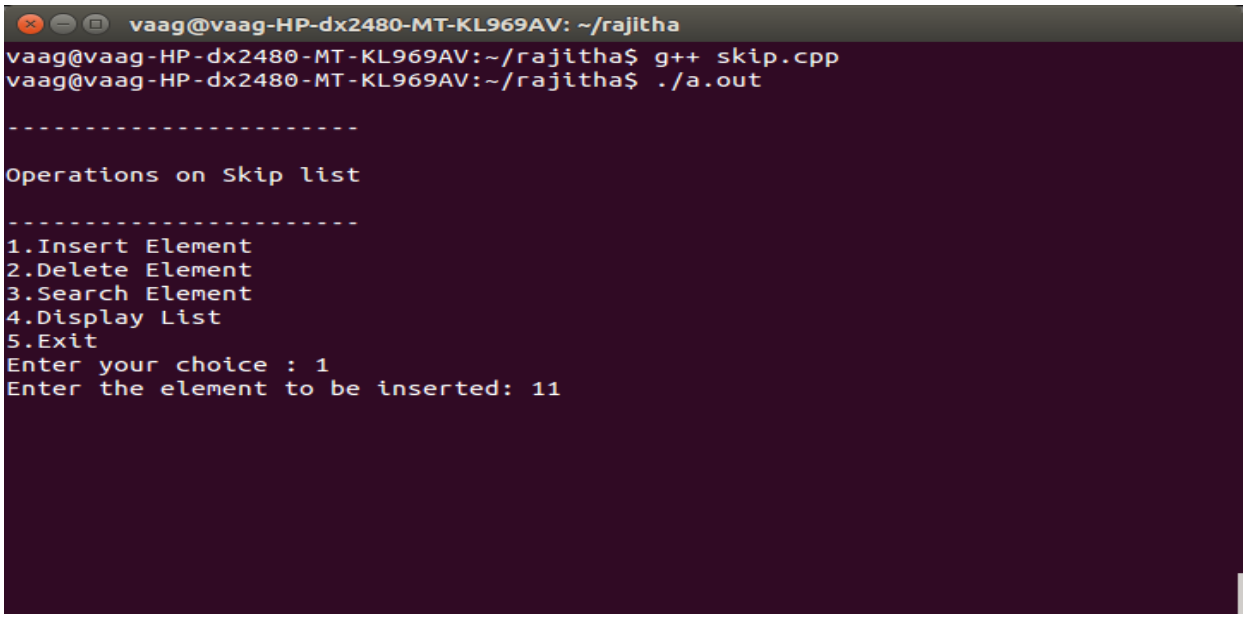
```
#include <iostream>
#include <stdlib.h>
#include <math.h>
#include <string.h>
using namespace std;
#define MAX_LEVEL 6
const float P = 0.5;
// Skip Node Declaration
struct snode
{
    int value;
    snode **forw;
    snode(int level, int &value)
    {
        forw = new snode * [level + 1];
        memset(forw, 0, sizeof(snode*) * (level + 1));
        this->value = value;
    }
    ~snode()
    {
        delete [] forw;
    }
};
// Skip List Declaration
struct skiplist
{
    snode *header;
    int value;
    int level;
    skiplist()
    {
        header = new snode(MAX_LEVEL, value);
        level = 0;
    }
    ~skiplist()
    {
        delete header;
    }
    void display();
    bool contains(int &);
    void insert_element(int &);
    void delete_element(int &);
};
int main()
{
    skiplist ss;
    int choice, n;
    while (1)
    {
        cout<<endl<<"----------------------"<<endl;
```

```
        cout<<endl<<"Operations on Skip list"<<endl;
        cout<<endl<<"-----------------------"<<endl;
        cout<<"1.Insert Element"<<endl;
        cout<<"2.Delete Element"<<endl;
        cout<<"3.Search Element"<<endl;
        cout<<"4.Display List "<<endl;
        cout<<"5.Exit "<<endl;
        cout<<"Enter your choice : ";
        cin>>choice;
        switch(choice)
        {
           case 1:cout<<"Enter the element to be inserted: ";
                  cin>>n;
                  ss.insert_element(n);
                  if(ss.contains(n))
                     cout<<"Element Inserted"<<endl;
                  break;
           case 2:cout<<"Enter the element to be deleted: ";
                  cin>>n;
                   if(!ss.contains(n))
                   {
                      cout<<"Element not found"<<endl;
                      break;
                   }
                   ss.delete_element(n);
                   if(!ss.contains(n))
                      cout<<"Element Deleted"<<endl;
                   break;
           case 3:cout<<"Enter the element to be searched: ";
                  cin>>n;
                   if(ss.contains(n))
                      cout<<"Element "<<n<<" is in the list"<<endl;
                   else
                      cout<<"Element not found"<<endl;
                   break;
           case 4:cout<<"The List is: ";
                   ss.display();
                   break;
           case 5:exit(1);
                  break;
           default:cout<<"Wrong Choice"<<endl;
        }
     }
     return 0;
}
//Random Value Generator
float frand()
{
   return (float) rand() / RAND_MAX;
}
```

```
// Random Level Generator
int random_level()
{
    static bool first = true;
    if(first)
    {
        srand((unsigned)time(NULL));
        first = false;
    }
    int lvl=(int)(log(frand())/log(1.-P));
    return lvl<MAX_LEVEL ? lvl : MAX_LEVEL;
}
// Insert Element in Skip List
void skiplist::insert_element(int &value)
{
    snode *x = header;
    snode *update[MAX_LEVEL + 1];
    memset(update,0,sizeof(snode*) * (MAX_LEVEL+1));
    for(int i=level;i>=0;i--)
    {
        while(x->forw[i]!=NULL && x->forw[i]->value<value)
        {
           x = x->forw[i];
        }
        update[i] = x;
    }
    x=x->forw[0];
    if(x==NULL || x->value!=value)
    {
        int lvl=random_level();
        if(lvl>level)
        {
           for(int i=level+1;i<=lvl;i++)
           {
               update[i] = header;
           }
           level=lvl;
        }
        x=new snode(lvl,value);
        for(int i=0;i<=lvl;i++)
        {
            x->forw[i]=update[i]->forw[i];
            update[i]->forw[i]=x;
        }
    }
}
// Delete Element from Skip List
void skiplist::delete_element(int &value)
{
    snode *x = header;
    snode *update[MAX_LEVEL + 1];
    memset(update,0,sizeof(snode*) * (MAX_LEVEL+1));
```

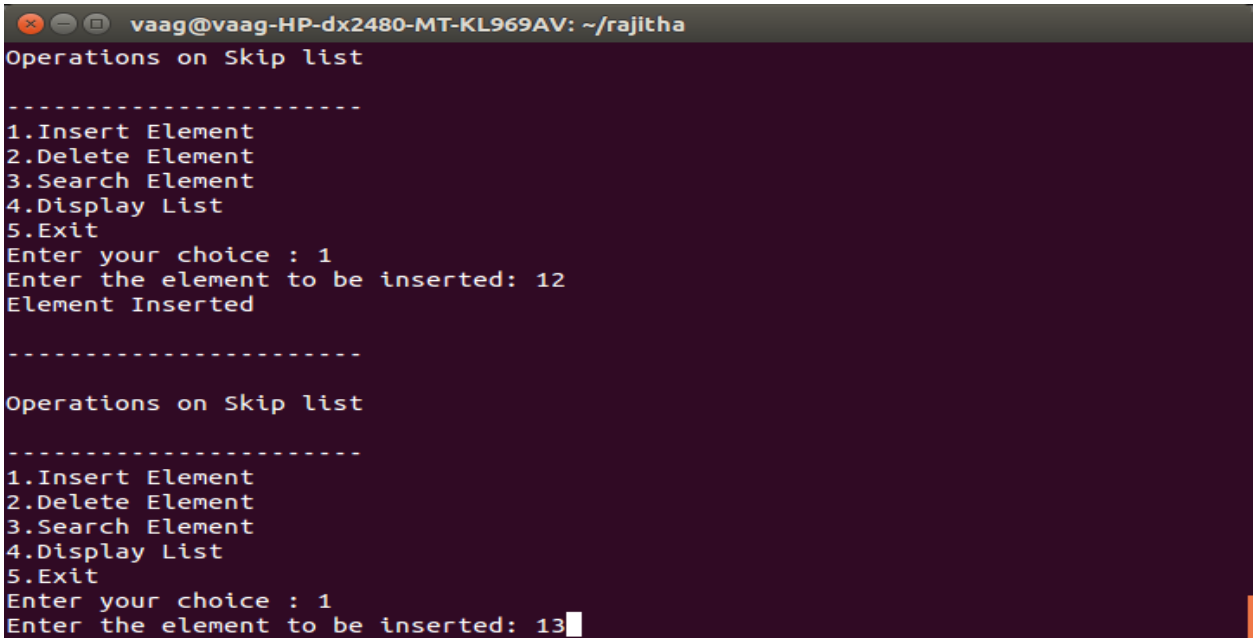
```
    for(int i=level;i>=0;i--)
    {
        while(x->forw[i]!=NULL && x->forw[i]->value < value)
        {
            x=x->forw[i];
        }
        update[i]=x;
    }
    x=x->forw[0];
    if(x->value==value)
    {
        for(int i=0;i<=level;i++)
        {
           if(update[i]->forw[i]!=x)
               break;
           update[i]->forw[i]=x->forw[i];
        }
        delete x;
        while(level>0 && header->forw[level]==NULL)
        {
           level--;
        }
    }
}
// Display Elements of Skip List
void skiplist::display()
{
    const snode *x=header->forw[0];
    while(x!=NULL)
    {
        cout<<x->value;
        x=x->forw[0];
        if(x!=NULL)
           cout << " - ";
    }
    cout<<endl;
}
// Search Elemets in Skip List
bool skiplist::contains(int &s_value)
{
    snode *x=header;
    for(int i=level;i>=0;i--)
    {
        while(x->forw[i]!=NULL && x->forw[i]->value < s_value)
        {
            x=x->forw[i];
        }
    }
    x=x->forw[0];
    return x!=NULL && x->value == s_value;
}
```

output:


vaag@vaag-HP-dx2480-MT-KL969AV: ~/rajitha
vaag@vaag-HP-dx2480-MT-KL969AV:~/rajitha$ g++ skip.cpp
vaag@vaag-HP-dx2480-MT-KL969AV:~/rajitha$ ./a.out

Operations on Skip list

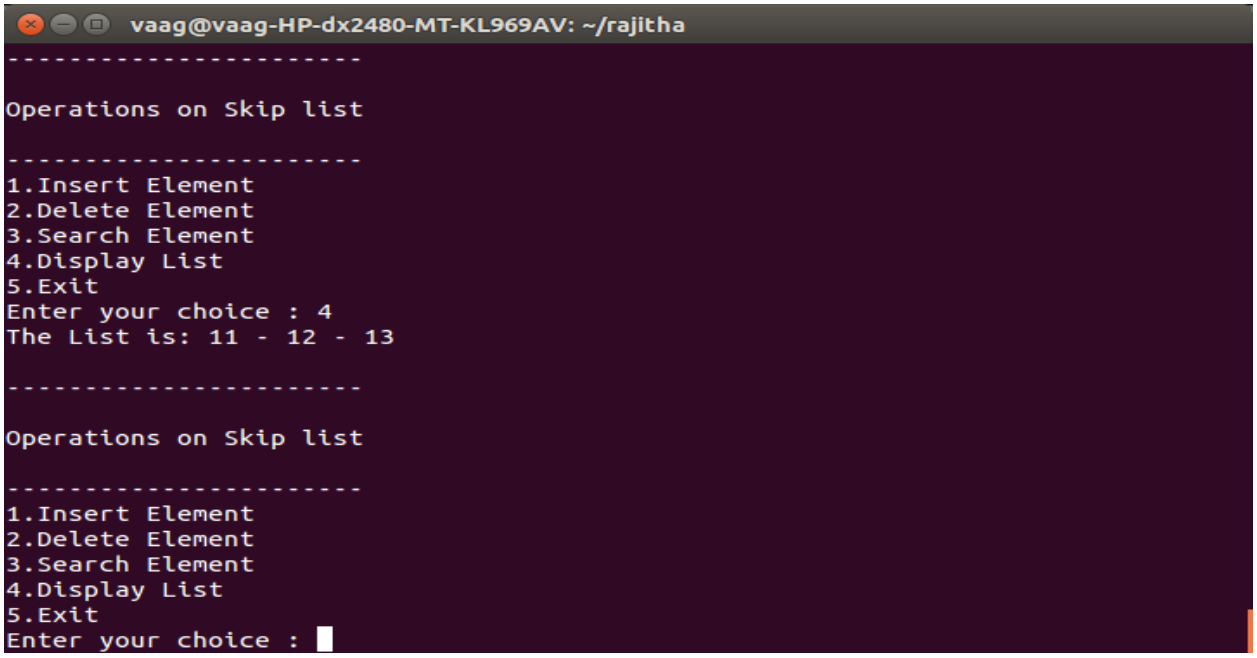
1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice : 1
Enter the element to be inserted: 11


vaag@vaag-HP-dx2480-MT-KL969AV: ~/rajitha
Operations on Skip list

1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice : 1
Enter the element to be inserted: 12
Element Inserted

Operations on Skip list

1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice : 1
Enter the element to be inserted: 13

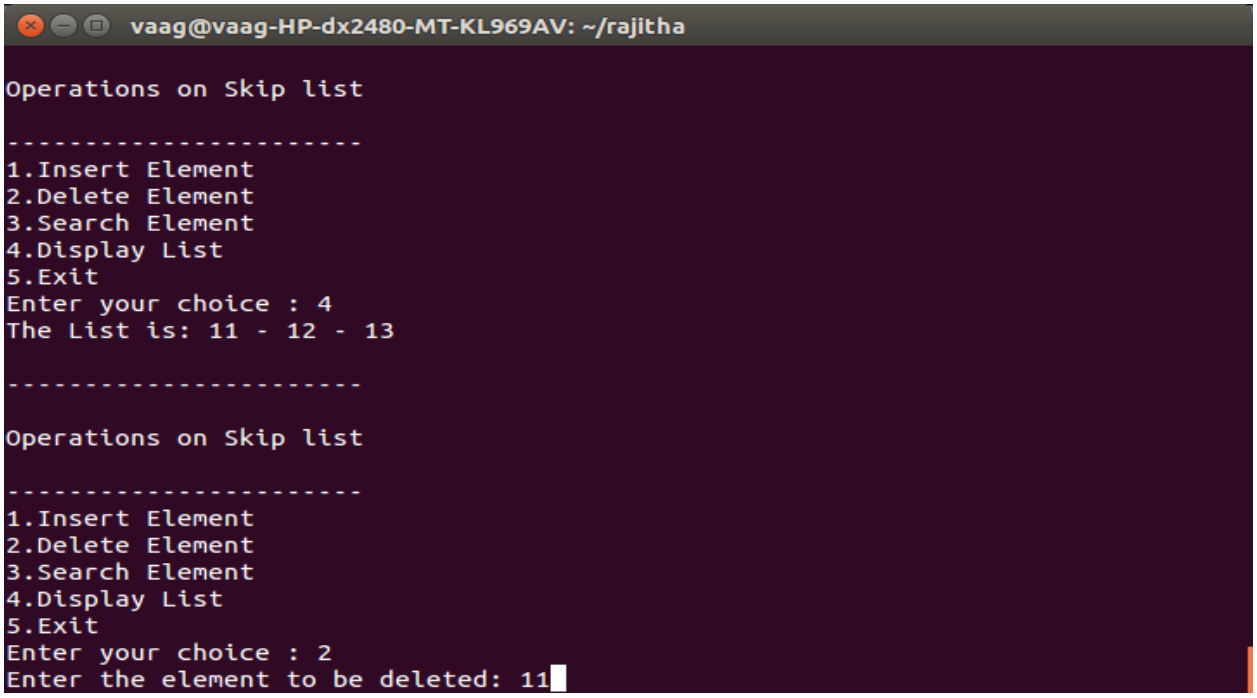

vaag@vaag-HP-dx2480-MT-KL969AV: ~/rajitha

Operations on Skip list

1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice : 4
The List is: 11 - 12 - 13

Operations on Skip list

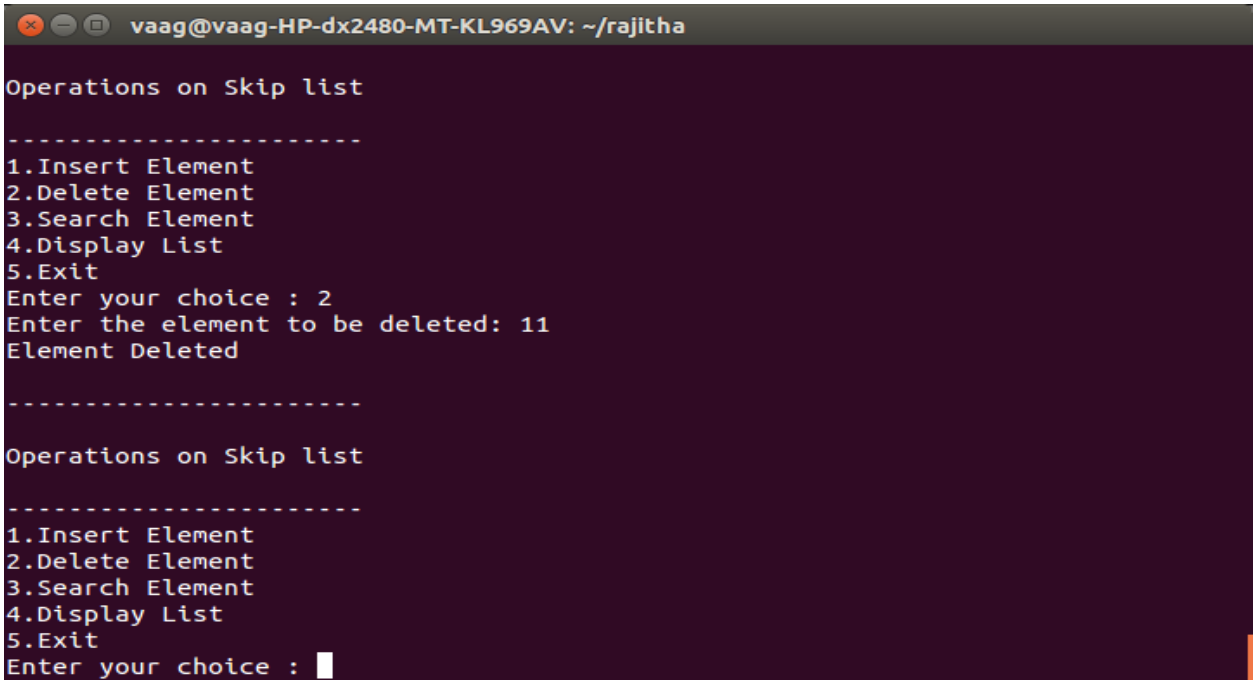
1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice :


vaag@vaag-HP-dx2480-MT-KL969AV: ~/rajitha
Operations on Skip list

1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice : 4
The List is: 11 - 12 - 13

Operations on Skip list

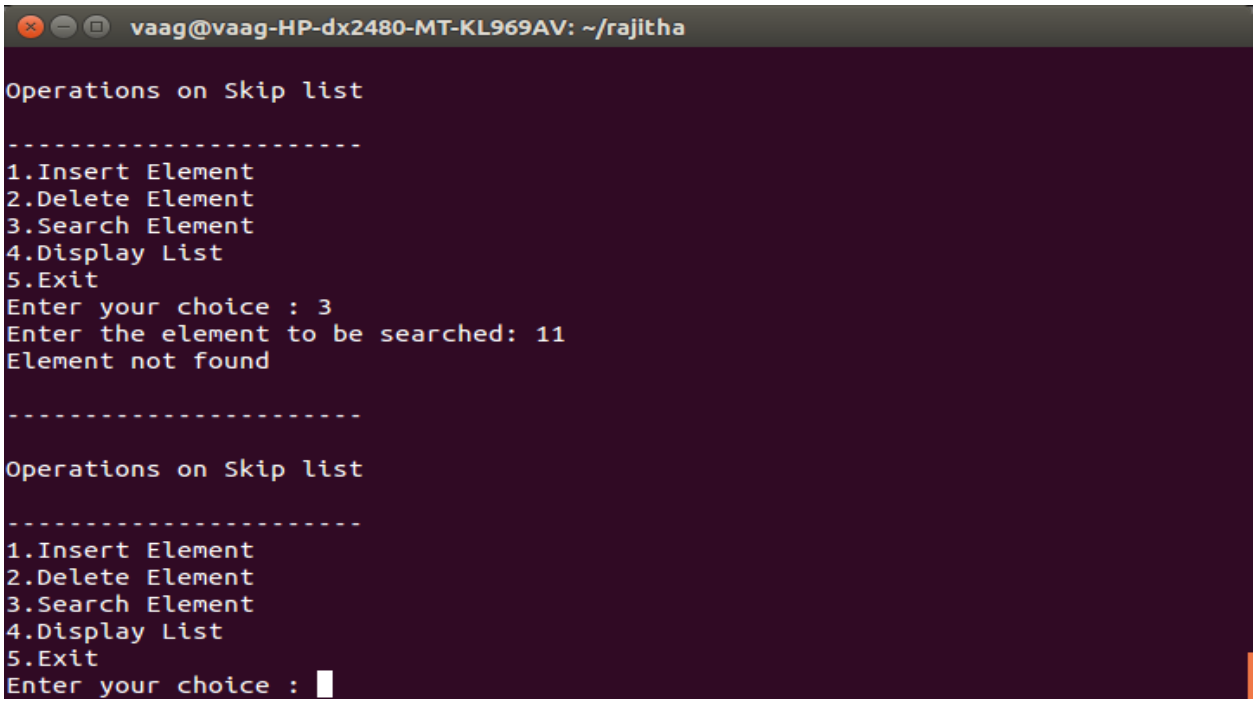
1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice : 2
Enter the element to be deleted: 11


vaag@vaag-HP-dx2480-MT-KL969AV: ~/rajitha
Operations on Skip list

1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice : 2
Enter the element to be deleted: 11
Element Deleted

Operations on Skip list

1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice :


vaag@vaag-HP-dx2480-MT-KL969AV: ~/rajitha
Operations on Skip list

1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice : 3
Enter the element to be searched: 11
Element not found

Operations on Skip list

1.Insert Element
2.Delete Element
3.Search Element
4.Display List
5.Exit
Enter your choice :

3. Write a C++ program for hashing with quadratic programming.

```
#include <iostream>
#include <stdlib.h>
#define MIN_TABLE_SIZE 10
using namespace std;
// Node Type Declaration
enum EntryType {Legitimate, Empty, Deleted};
// Node Declaration
struct HashNode
{
    int element;
    enum EntryType info;
};
// Table Declaration
struct HashTable
{
    int size;
    HashNode *table;
};
// Returns whether n is prime or not
bool isPrime(int n)
{
    if(n==2 || n==3)
            return true;
    if(n==1 || n%2==0)
            return false;
    for(int i=3;i*i<=n;i+=2)
            if(n%i==0)
               return false;
    return true;
}
// Finding next prime size of the table
int nextPrime(int n)
{
    if(n<=0)
            n==3;
    if(n%2==0)
            n++;
    for(;!isPrime(n);n+=2);
        return n;
}
// Function To Generate Hash
int HashFunc(int key, int size)
{
    return key % size;
}
//Function to Initialize Table
HashTable *initializeTable(int size)
{
    HashTable *htable;
```

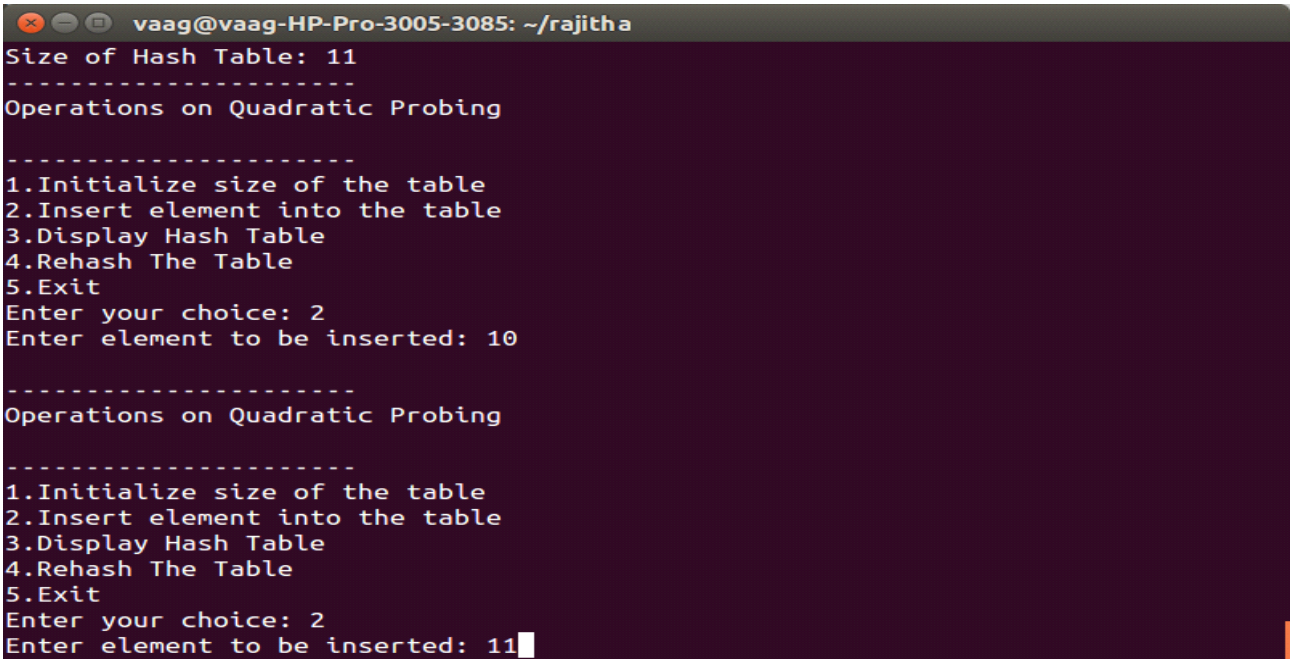
```
    if(size<MIN_TABLE_SIZE)
    {
        cout<<"Table Size Too Small"<<endl;
        return NULL;
    }
    htable=new HashTable;
    if(htable==NULL)
    {
        cout<<"Out of Space"<<endl;
        return NULL;
    }
    htable->size=nextPrime(size);
    htable->table=new HashNode[htable->size];
    if(htable->table==NULL)
    {
        cout<<"Table Size Too Small"<<endl;
        return NULL;
    }
    for(int i=0;i<htable->size;i++)
    {
        htable->table[i].info=Empty;
        htable->table[i].element=NULL;
    }
    return htable;
}
//Function to Find Element at a key
int Find(int key, HashTable *htable)
{
    int pos = HashFunc(key, htable->size);
    int collisions = 0;
    while(htable->table[pos].info!=Empty && htable->table[pos].element!=key)
    {
        pos=pos+2* ++collisions -1;
        if(pos>=htable->size)
           pos=pos-htable->size;
    }
    return pos;
}
//Function to Insert Element into a key
void Insert(int key, HashTable *htable)
{
    int pos=Find(key,htable);
    if(htable->table[pos].info!=Legitimate)
    {
        htable->table[pos].info=Legitimate;
        htable->table[pos].element=key;
    }
}
```

```
//Function to Rehash the Table
HashTable *Rehash(HashTable *htable)
{
    int size=htable->size;
    HashNode *table=htable->table;
    htable=initializeTable(2 * size);
    for(int i=0;i<size;i++)
    {
            if(table[i].info==Legitimate)
                Insert(table[i].element, htable);
    }
    free(table);
    return htable;
}
//Function to Retrieve Hash Table
void Retrieve(HashTable *htable)
{
    for(int i=0;i<htable->size;i++)
    {
            int value=htable->table[i].element;
            if(!value)
               cout<<"Position: "<<i + 1<<" Element: Null"<<endl;
            else
               cout<<"Position: "<<i + 1<<" Element: "<<value<<endl;
    }
}
//Main Contains Menu
int main()
{
    int value,size,pos,i = 1;
    int choice;
    HashTable *htable;
    while(1)
    {
            cout<<"\n----------------------"<<endl;
            cout<<"Operations on Quadratic Probing"<<endl;
            cout<<"\n----------------------"<<endl;
            cout<<"1.Initialize size of the table"<<endl;
            cout<<"2.Insert element into the table"<<endl;
            cout<<"3.Display Hash Table"<<endl;
            cout<<"4.Rehash The Table"<<endl;
            cout<<"5.Exit"<<endl;
            cout<<"Enter your choice: ";
            cin>>choice;
            switch(choice)
            {
                case 1:cout<<"Enter size of the Hash Table: ";
               cin>>size;
                        htable=initializeTable(size);
               cout<<"Size of Hash Table: "<<nextPrime(size);
                        break;
```

```
                case 2:if(i>htable->size)
                        {
                                cout<<"Table is Full, Rehash the table"<<endl;
                                continue;
                         }
                        cout<<"Enter element to be inserted: ";
                          cin>>value;
                          Insert(value, htable);
                          i++;
                          break;
                case 3:Retrieve(htable);
                          break;
                case 4:htable=Rehash(htable);
                          break;
                case 5:exit(1);
               default:cout<<"\nEnter correct option\n";
        }
    }
    return 0;
}
```

output:

```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ ./a.out

----------------------
Operations on Quadratic Probing

----------------------
1.Initialize size of the table
2.Insert element into the table
3.Display Hash Table
4.Rehash The Table
5.Exit
Enter your choice: 1
Enter size of the Hash Table: 10
Size of Hash Table: 11
----------------------
Operations on Quadratic Probing

----------------------
1.Initialize size of the table
2.Insert element into the table
3.Display Hash Table
4.Rehash The Table
5.Exit
Enter your choice:
```

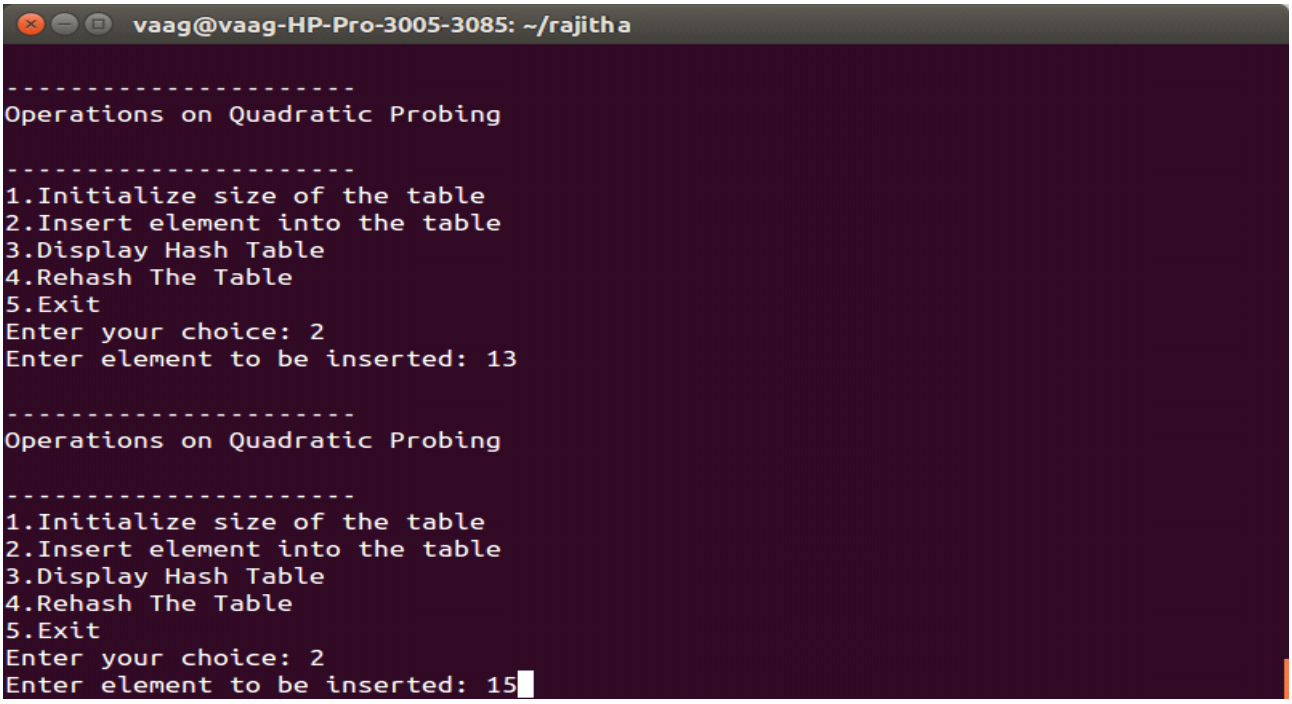

vaag@vaag-HP-Pro-3005-3085: ~/rajitha
Size of Hash Table: 11

Operations on Quadratic Probing

1.Initialize size of the table
2.Insert element into the table
3.Display Hash Table
4.Rehash The Table
5.Exit
Enter your choice: 2
Enter element to be inserted: 10

Operations on Quadratic Probing

1.Initialize size of the table
2.Insert element into the table
3.Display Hash Table
4.Rehash The Table
5.Exit
Enter your choice: 2
Enter element to be inserted: 11

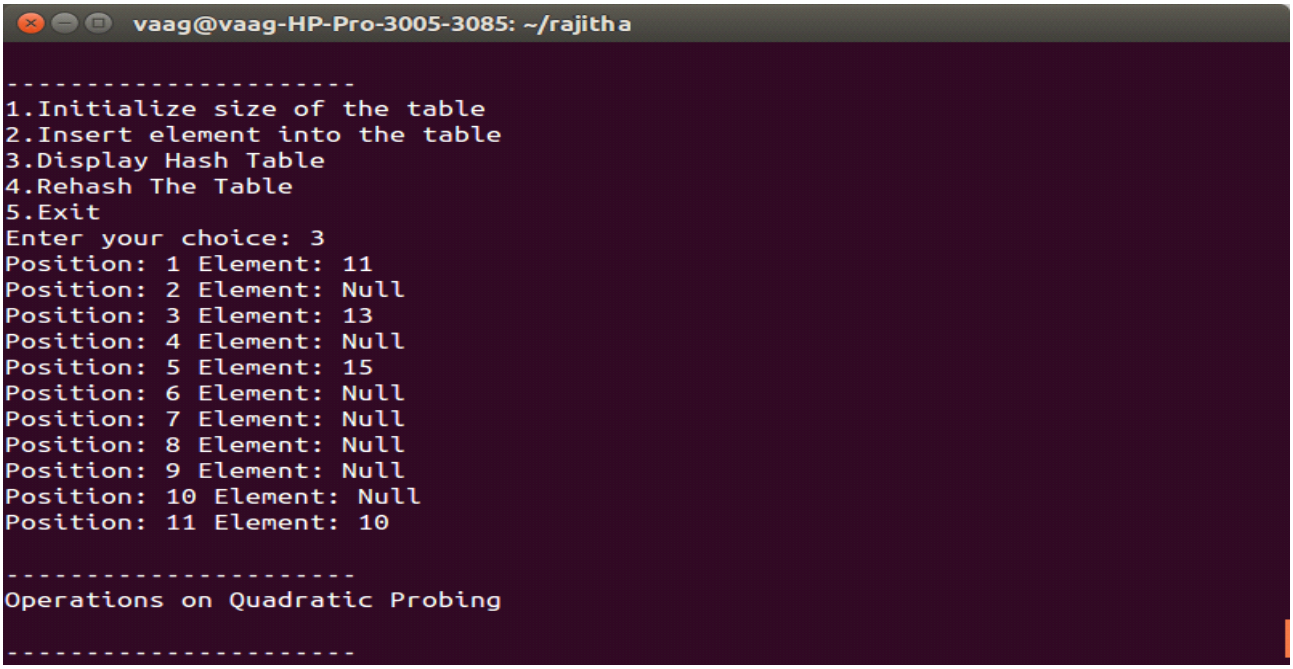

vaag@vaag-HP-Pro-3005-3085: ~/rajitha

Operations on Quadratic Probing

1.Initialize size of the table
2.Insert element into the table
3.Display Hash Table
4.Rehash The Table
5.Exit
Enter your choice: 2
Enter element to be inserted: 13

Operations on Quadratic Probing

1.Initialize size of the table
2.Insert element into the table
3.Display Hash Table
4.Rehash The Table
5.Exit
Enter your choice: 2
Enter element to be inserted: 15


vaag@vaag-HP-Pro-3005-3085: ~/rajitha
1.Initialize size of the table
2.Insert element into the table
3.Display Hash Table
4.Rehash The Table
5.Exit
Enter your choice: 3
Position: 1 Element: 11
Position: 2 Element: Null
Position: 3 Element: 13
Position: 4 Element: Null
Position: 5 Element: 15
Position: 6 Element: Null
Position: 7 Element: Null
Position: 8 Element: Null
Position: 9 Element: Null
Position: 10 Element: Null
Position: 11 Element: 10
Operations on Quadratic Probing

4. C++ programs using class templates to implement the stack using an array.

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
template<class t>
class stack
{
    int top,n;
    t a[50];
    public:
            stack(int m)
            {
               top=-1;
               n=m;
            }
    void push();
    void pop();
    void display();
};
template<class t>
void stack<t>::push()
{
    t ele;
    if(top>=n-1)
    {
            cout<<"stack is over flow\n";
            return;
    }
    top++;
    cout<<"enter the element \n";
    cin>>ele;
    a[top]=ele;
}
template<class t>
void stack<t>::pop()
{
    t ele;
    if(top==-1)
    {
            cout<<"stack is underflow\n";
            return;
    }
    ele=a[top];
    top--;
    cout<<"the deleted element is:"<<ele;
}
```

```
template<class t>
void stack<t>::display()
{
    int i;
    if(top==-1)
    {
            cout<<"stack is underflow\n";
            return;
    }
    cout<<"the elements are:\n";
    for(i=top;i>=0;i--)
            cout<<a[i]<<" ";
}
int main()
{
    int n,ch;
    clrscr();
    cout<<"enter the size of the stack:\n";
    cin>>n;
    stack<int>s(n);
    while(1)
    {
            cout<<"\n menu 1.push. 2.pop 3.display 4.exit\n";
            cout<<"enter ur choice:";
            cin>>ch;
            switch(ch)
            {
               case 1:s.push();
                        break;
               case 2:s.pop();
                        break;
               case 3:s.display();
                        break;
               case 4:exit(0);
                        break;
              default:cout<<"invalid choice";
                        break;
            }
    }
}
```

output:

```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ g++ stackarray.cpp
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ ./a.out
enter the size of the stack:
5

 menu 1.push. 2.pop 3.display 4.exit
enter ur choice:1
enter the element
1

 menu 1.push. 2.pop 3.display 4.exit
enter ur choice:1
enter the element
2

 menu 1.push. 2.pop 3.display 4.exit
enter ur choice:1
enter the element
3

 menu 1.push. 2.pop 3.display 4.exit
enter ur choice:
```

```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
3

 menu 1.push. 2.pop 3.display 4.exit
enter ur choice:3
the elements are:
3 2 1
 menu 1.push. 2.pop 3.display 4.exit
enter ur choice:2
the deleted element is:3
 menu 1.push. 2.pop 3.display 4.exit
enter ur choice:3
the elements are:
2 1
 menu 1.push. 2.pop 3.display 4.exit
enter ur choice:1
enter the element
4

 menu 1.push. 2.pop 3.display 4.exit
enter ur choice:3
the elements are:
4 2 1
 menu 1.push. 2.pop 3.display 4.exit
enter ur choice:
```

4. C++ programs using class templates to implement the queue using an array.

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
template<class t>
class queue
{
    int rear,front,n;
    t a[50];
    public:
        queue(int m)
        {
            rear=front=-1;
            n=m;
        }
        void enque();
        void deque();
        void display();
};
template<class t>
void queue<t>::enque()
{
    t ele;
    if(rear>=n-1)
    {
        cout<<"queue is overflow\n";
        return;
    }
    cout<<"enter the element\n";
    cin>>ele;
    if(front==-1)
        rear=front=0;
    else
        rear++;
    a[rear]=ele;
}
template<class t>
void queue<t>::deque()
{
    t ele;
    if(rear==-1)
    {
        cout<<"queue is underflow\n";
        return;
    }
    ele=a[front];
    if(rear==front)
        rear=front=-1;
    else
        front++;
```

```
    cout<<"the deleted element is:"<<ele;
}
template<class t>
void queue<t>::display()
{
    int i;
    if(rear==-1)
    {
            cout<<"queue is underflow\n";
            return;
    }
    for(i=front;i<=rear;i++)
            cout<<a[i]<<" ";
    cout<<"\n";
}
int main()
{
    int ch,n;
    clrscr();
    cout<<"enter the size of the queue\n";
    cin>>n;
    queue<int>q(n);
    while(1)
    {
            cout<<"\n menu 1eenque 2. deque 3.display 4.exit\n";
            cout<<"enter ur choice:";
            cin>>ch;
            switch(ch)
            {
                case 1:q.enque();
                        break;
                case 2:q.deque();
                        break;
                case 3:q.display();
                        break;
                case 4:exit(0);
                        break;
               default:cout<<"invalid option\n";
                        break;
            }
    }
}
```

output:

```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ g++ queuearray.cpp
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ ./a.out
enter the size of the queue
5

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:1
enter the element
1

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:1
enter the element
2

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:1
enter the element
3

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:
```

```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
3

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:3
1 2 3

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:2
the deleted element is:1
 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:3
2 3

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:1
enter the element
4

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:3
2 3 4

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:
```

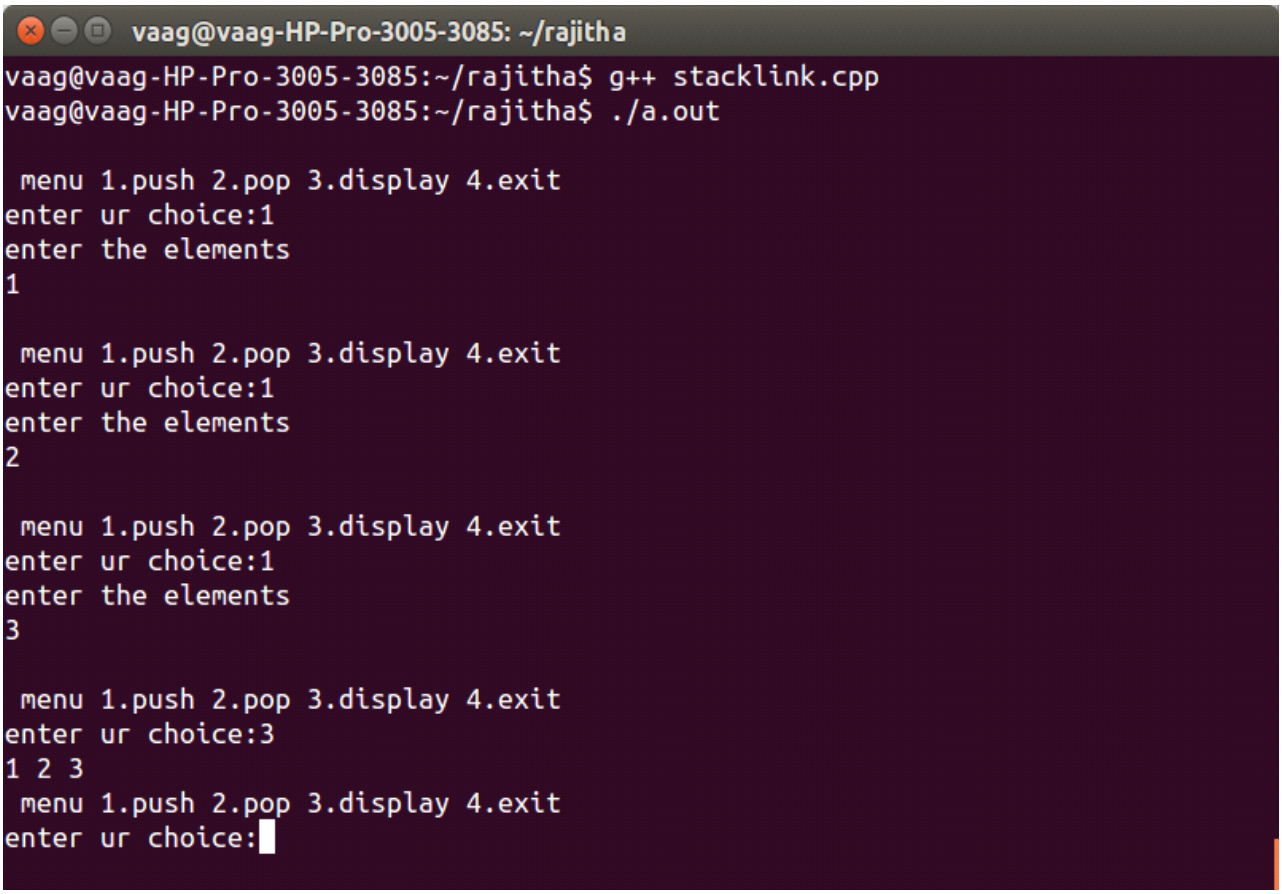
5. Write C++ programs using class templates to implement the stack using a singly linked list.

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
#include<alloc.h>
template<class t>
class llist
{
   struct node
   {
         t data;
         node *link;
   }*top,*head,*temp,*temp1;
   public:
         llist();
         void push();
         void pop();
         void display();
};
template<class t>
llist<t>::llist()
{
   top=NULL;
   head->link=NULL;
}
template<class t>
void llist<t>::push()
{
   t ele;
   cout<<"enter the elements\n";
   cin>>ele;
   temp=(struct node*)malloc(sizeof(struct node));
   if(temp==NULL)
   {
         cout<<"memory allocation error\n";
         return;
   }
   temp->data=ele;
   if(top==NULL)
   {
         head->link=temp;
         temp->link=NULL;
   }
   else
   {
         top->link=temp;
         temp->link=NULL;
   }
   top=temp;
}
```

```
template<class t>
void llist<t>::pop()
{
    t ele;
    if(top==NULL)
    {
            cout<<"linked list is empty\n";
            return;
    }
    for(temp1=head;temp1!=top;temp1=temp1->link)
    {
            temp=temp1;
    }
    temp->link=NULL;
    ele=temp1->data;
    free(temp1);
    top=temp;
    cout<<"the deleted element is:"<<ele;
}
template<class t>
void llist<t>::display()
{
    if(top==NULL)
    {
            cout<<"linked list is empty\n";
            return;
    }
    for(temp=head->link;temp!=top->link;temp=temp->link)
    {
            cout<<temp->data<<" ";
    }
}
int main()
{
    int ch;
    llist<int>l;
    clrscr();
    while(1)
    {
            cout<<"\n menu 1.push 2.pop 3.display 4.exit\n:";
            cout<<"enter ur choice:";
            cin>>ch;
            switch(ch)
            {
                case 1:l.push();
                        break;
                case 2:l.pop();
                        break;
```

```
            case 3:l.display();
                    break;
            case 4:exit(0);
                    break;
        }
    }
}
```

output:



```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ g++ stacklink.cpp
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ ./a.out

 menu 1.push 2.pop 3.display 4.exit
enter ur choice:1
enter the elements
1

 menu 1.push 2.pop 3.display 4.exit
enter ur choice:1
enter the elements
2

 menu 1.push 2.pop 3.display 4.exit
enter ur choice:1
enter the elements
3

 menu 1.push 2.pop 3.display 4.exit
enter ur choice:3
1 2 3
 menu 1.push 2.pop 3.display 4.exit
enter ur choice:
```

```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
 menu 1.push 2.pop 3.display 4.exit
enter ur choice:1
enter the elements
3

 menu 1.push 2.pop 3.display 4.exit
enter ur choice:3
1 2 3
 menu 1.push 2.pop 3.display 4.exit
enter ur choice:2
the deleted element is:3
 menu 1.push 2.pop 3.display 4.exit
enter ur choice:3
1 2
 menu 1.push 2.pop 3.display 4.exit
enter ur choice:1
enter the elements
4

 menu 1.push 2.pop 3.display 4.exit
enter ur choice:3
1 2 4
 menu 1.push 2.pop 3.display 4.exit
enter ur choice:
```

5. Write C++ programs using class templates to implement the queue using a singly linked list.

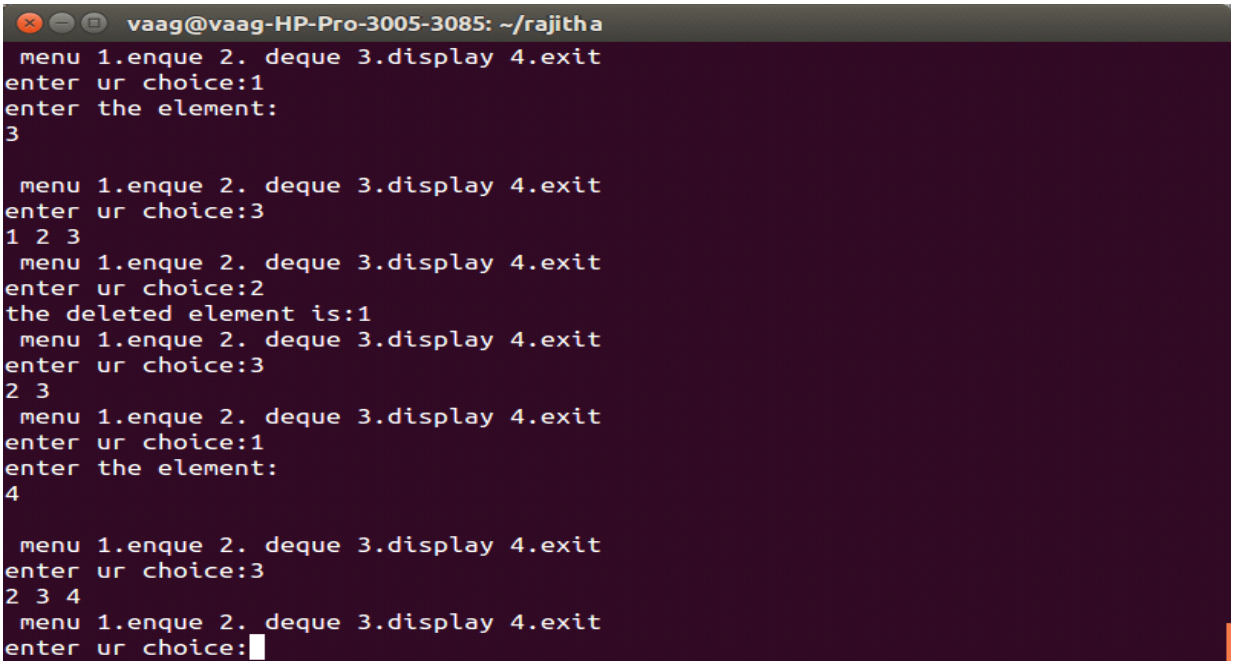
```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
template<class t>
class queue
{
   struct node
   {
         t data;
         node *link;
   }*head,*temp,*front,*rear;
   public:
        queue();
        void enque();
        void deque();
        void display();
};
template<class t>
void queue<t>::queue()
{
   head->link=NULL;
   rear=NULL;
   front=NULL;
}
template<class t>
void queue<t>::enque()
{
   t ele;
   cout<<"enter the element:\n";
   cin>>ele;
   temp=(struct node*)malloc(sizeof(struct node));
   if(temp==NULL)
   {
         cout<<"memory allocation error\n";
         return;
   }
   if(front==NULL)
   {
         temp->data=ele;
         head->link=temp;
         temp->link=NULL;
         front=rear=temp;
   }
   else
   {
         temp->data=ele;
         rear->link=temp;
         temp->link=NULL;
         rear=temp;
```

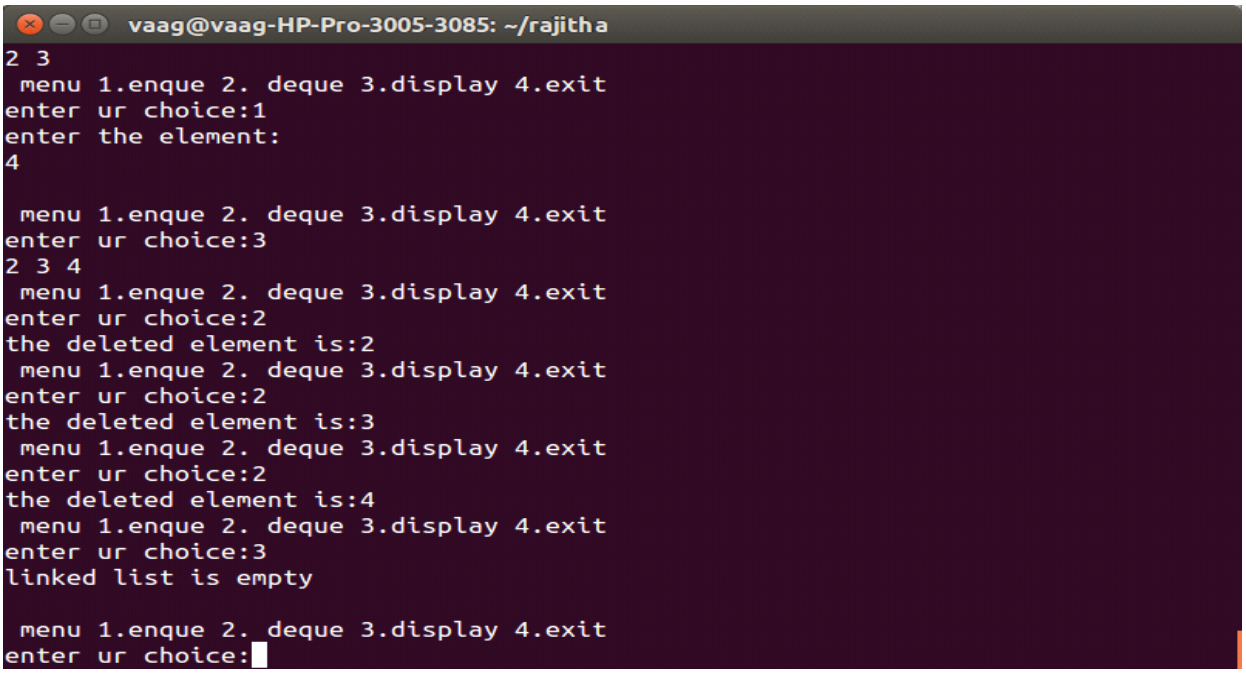
```
    }
}
template<class t>
void queue<t>::deque()
{
    t ele;
    if(front==NULL)
    {
            cout<<"linked list is empty\n";
            return;
    }
    if(front==rear)
    {
            ele=front->data;
            front=rear=NULL;
    }
    else
    {
            ele=front->data;
            temp=front;
            front=front->link;
            free(temp);
    }
    cout<<"the deleted element is:"<<ele;
}
template<class t>
void queue<t>::display()
{
    if(rear==NULL)
    {
            cout<<"linked list is empty\n";
            return;
    }
    for(temp=front;temp!=rear->link;temp=temp->link)
    {
            cout<<temp->data<<" ";
    }
}
int main()
{
    int ch;
    clrscr();
    queue<int>q;
    while(1)
    {
            cout<<"\n menu 1eenque 2. deque 3.display 4.exit\n";
            cout<<"enter ur choice:";
            cin>>ch;
```

```
        switch(ch)
        {
           case 1:q.enque();
                  break;
           case 2:q.deque();
                  break;
           case 3:q.display();
                  break;
           case 4:exit(0);
                  break;
          default:cout<<"invalid option\n";
                  break;
        }
   }
}
```

output:

```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ g++ queuelink.cpp
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ ./a.out

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:1
enter the element:
1

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:1
enter the element:
2

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:1
enter the element:
3

 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:3
1 2 3
 menu 1.enque 2. deque 3.display 4.exit
enter ur choice:
```


vaag@vaag-HP-Pro-3005-3085: ~/rajitha
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:1
enter the element:
3
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:3
1 2 3
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:2
the deleted element is:1
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:3
2 3
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:1
enter the element:
4
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:3
2 3 4
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:


vaag@vaag-HP-Pro-3005-3085: ~/rajitha
2 3
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:1
enter the element:
4
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:3
2 3 4
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:2
the deleted element is:2
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:2
the deleted element is:3
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:2
the deleted element is:4
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:3
linked list is empty
menu 1.enque 2. deque 3.display 4.exit
enter ur choice:

6. Write C++ programs using class templates to implement the deque (double ended queue) ADT using an array.

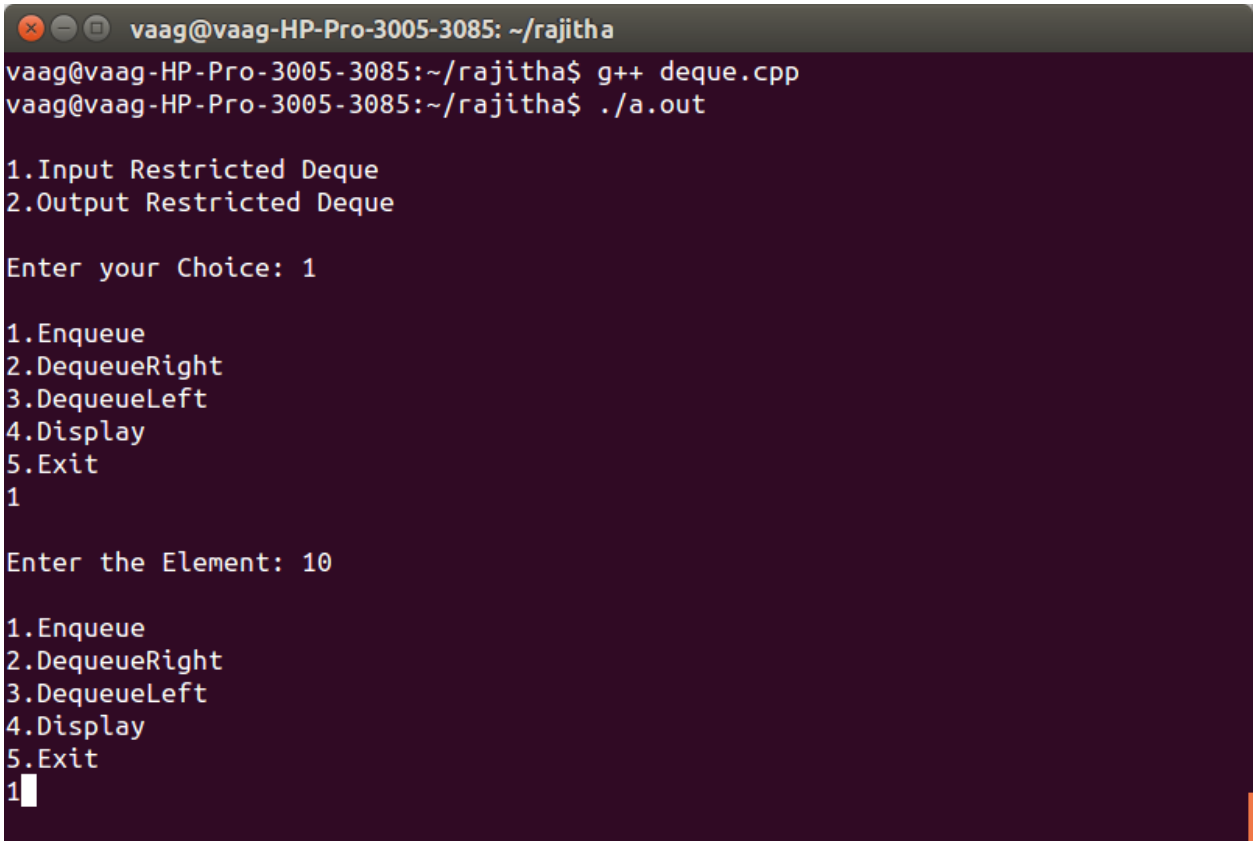
```
#include <iostream>
#define MAX 20
using namespace std;
struct queue
{
   int x[MAX];
   int front;
   int rear;
}q;
void enqueueRight(int);
void display();
int dequeueLeft();
void enqueueLeft(int);
int dequeueRight();
int main()
{
   q.front=-1;
   q.rear=-1;
   int ch,x,flag=1,c2;
   cout<<"\n1.Input Restricted Deque\n2.Output Restricted Deque\n";
   cout<<"\nEnter your Choice: ";
   cin>>c2;
   if(c2==1)
   {
    while(flag)
    {
     cout<<"\n1.Enqueue\n2.DequeueRight\n3.DequeueLeft\n4.Display\n5.Exit\n";
     cin>>ch;
     switch(ch)
     {
        case 1:cout<<"\nEnter the Element: ";
               cin>>x;
               enqueueRight(x);
               break;
        case 2:x=dequeueRight();
               cout<<"\nRemoved "<<x<<" from the Dequeue\n";
               break;
        case 3:x=dequeueLeft();
               cout<<"\nRemoved "<<x<<" form the Dequeue\n";
               break;
        case 4:display();
               break;
        case 5:flag=0;
               break;
        default:cout<<"\nWrong choice!!! Try Again.\n";
     }
    }
   }
```

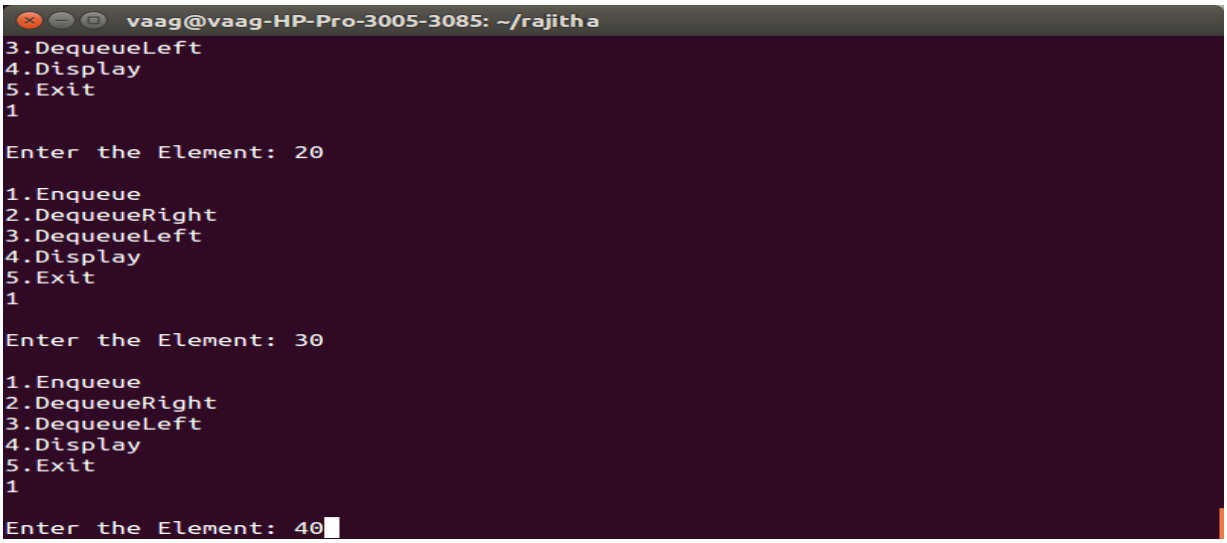
```
    else if(c2==2)
    {
     while(flag)
     {
      cout<<"\n1.EnqueueLeft\n2.EnqueueRight\n3.Dequeue\n4.Display\n5.Exit\n";
      cin>>ch;
      switch(ch)
      {
        case 1:cout<<"\nEnter the Element: ";
               cin>>x;
               enqueueLeft(x);
               break;
        case 2:cout<<"\nEnter the Element: ";
               cin>>x;
               enqueueRight(x);
               break;
        case 3:x=dequeueLeft();
               cout<<"\nRemoved "<<x<<" from the Dequeue\n";
               break;
        case 4:display();
               break;
        case 5:flag=0;
               break;
       default:cout<<"\nWrong Choice!!! Try Again.\n";
      }
     }
    }
    else
    {
      cout<<"\nWrong Choice!!!\n";
    }
    return 0;
}
void enqueueRight(int x)
{
    if(q.rear==MAX)
    {
       cout<<"\nDequeue Full from Right\n";
    }
    else
    {
       q.x[++q.rear]=x;
       if(q.front==-1)
       {
          q.front=0;
       }
    }
}
```

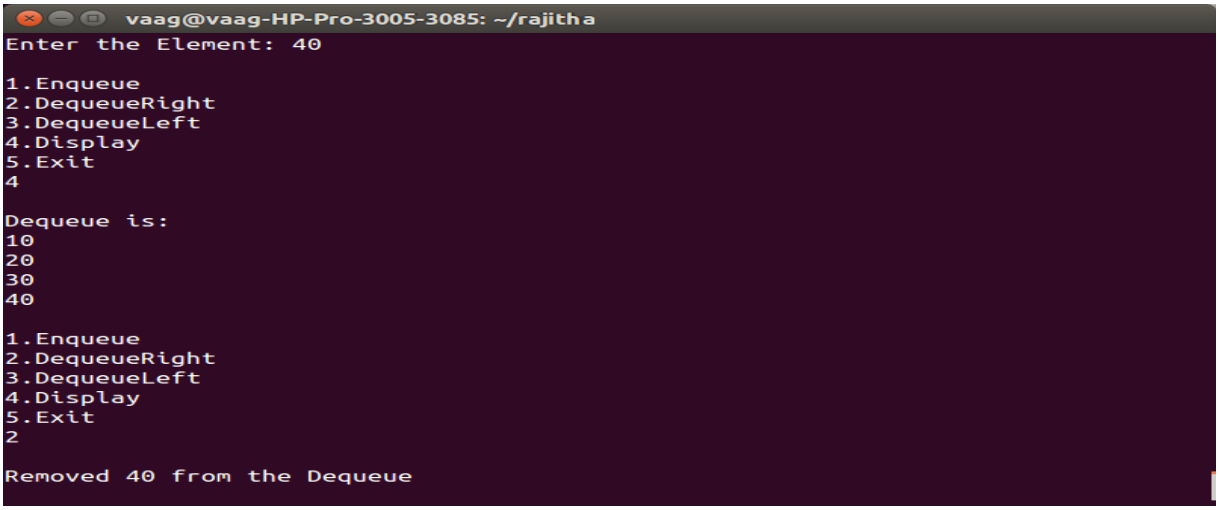
```
void enqueueLeft(int x)
{
   if(q.rear==-1 && q.front==-1)
   {
      enqueueRight(x);
   }
   else if(q.front==0)
   {
      cout<<"\nDequeue Full from Left\n";
   }
   else
   {
      q.x[--q.front]=x;
   }
}
int dequeueLeft()
{
   int x;
   if(q.rear==-1 && q.front==-1)
   {
      cout<<"\nDequeue Empty!!!\n";
   }
   else if(q.front==q.rear)
   {
      x=q.x[q.front];
      q.front=q.rear=-1;
      return x;
   }
   else
   {
      return q.x[q.front++];
   }
}
int dequeueRight()
{
   int x;
   if(q.rear==-1 && q.front==-1)
   {
      cout<<"\nDequeue Empty!!!\n";
   }
   else if(q.front==q.rear)
   {
      x=q.x[q.front];
      q.front=q.rear=-1;
      return x;
   }
   else
   {
      return q.x[q.rear--];
   }
}
```

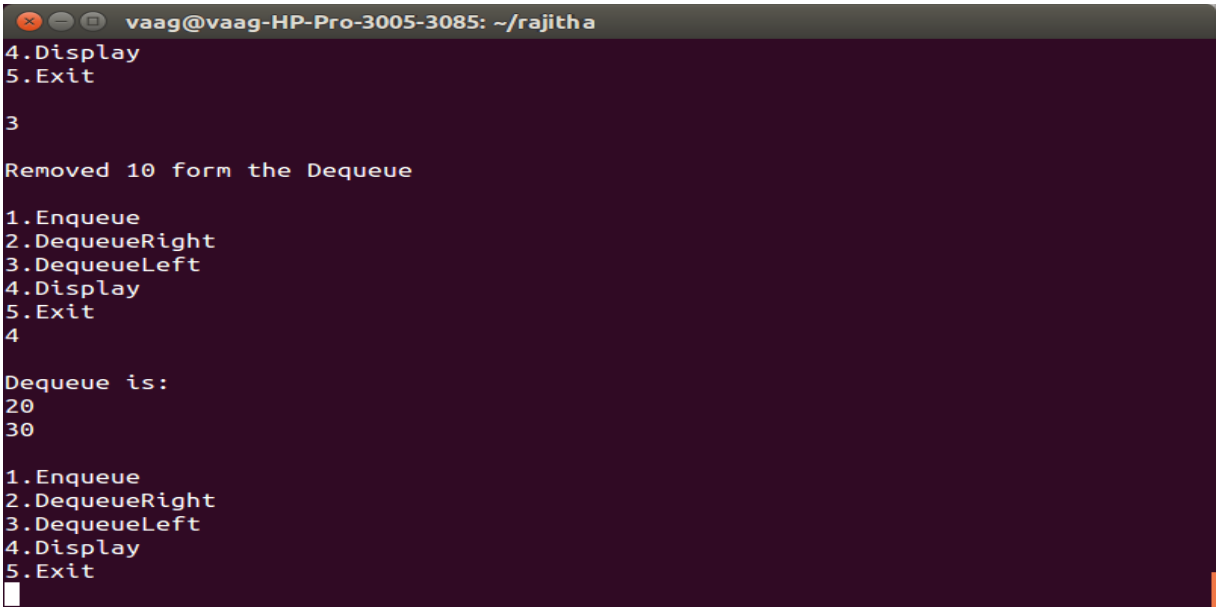
```
void display()
{
   int i;
   if(q.front==-1&&q.rear==-1)
   {
      cout<<"\nDequeue Empty!!!";
   }
   else
   {
     cout<<"\nDequeue is:\n";
     for(i=q.front;i<=q.rear;i++)
     {
       cout<<q.x[i]<<"\n";
     }
   }
}
```

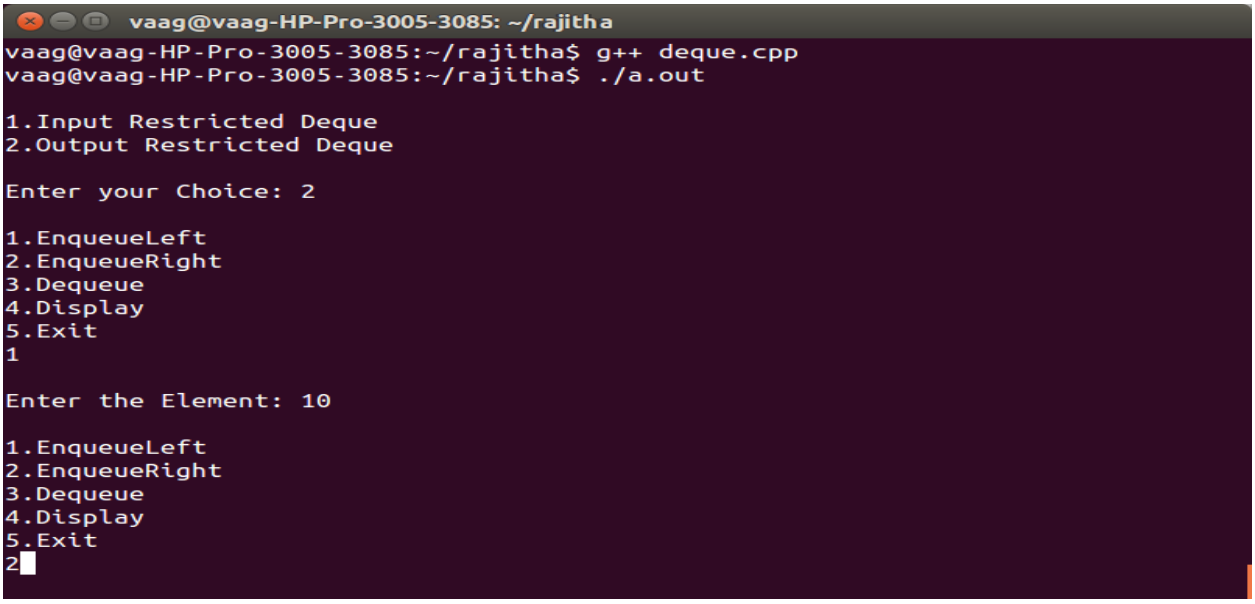
output:

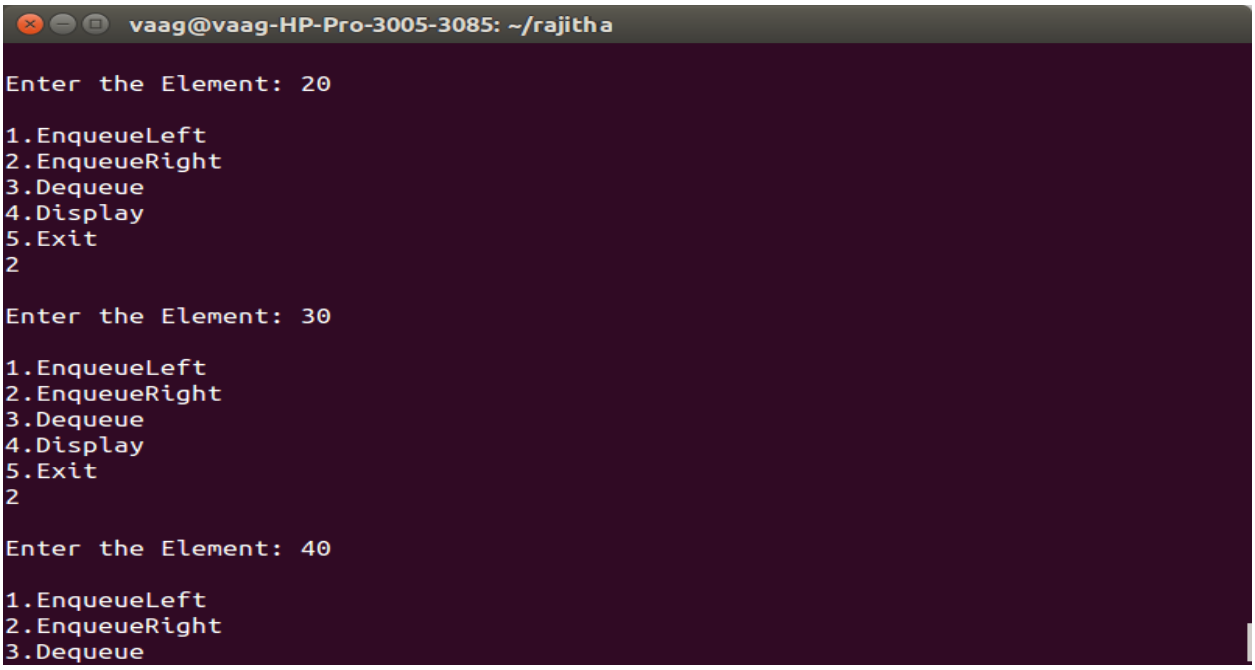


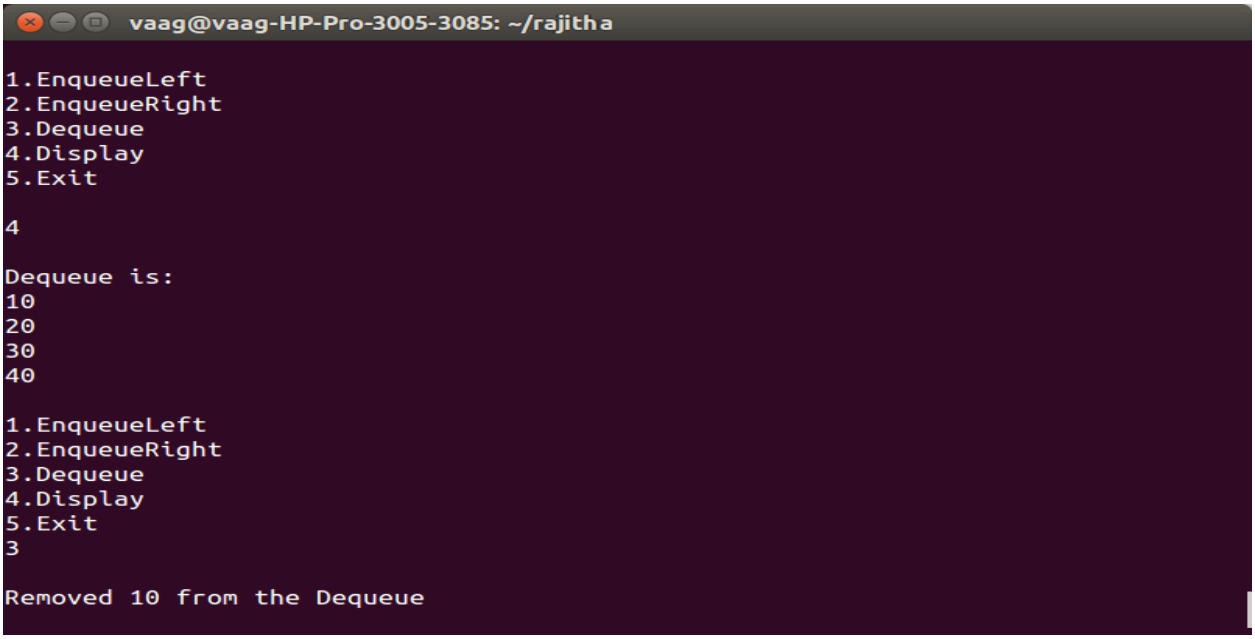

vaag@vaag-HP-Pro-3005-3085: ~/rajitha
3.DequeueLeft
4.Display
5.Exit
1
Enter the Element: 20
1.Enqueue
2.DequeueRight
3.DequeueLeft
4.Display
5.Exit
1
Enter the Element: 30
1.Enqueue
2.DequeueRight
3.DequeueLeft
4.Display
5.Exit
1
Enter the Element: 40

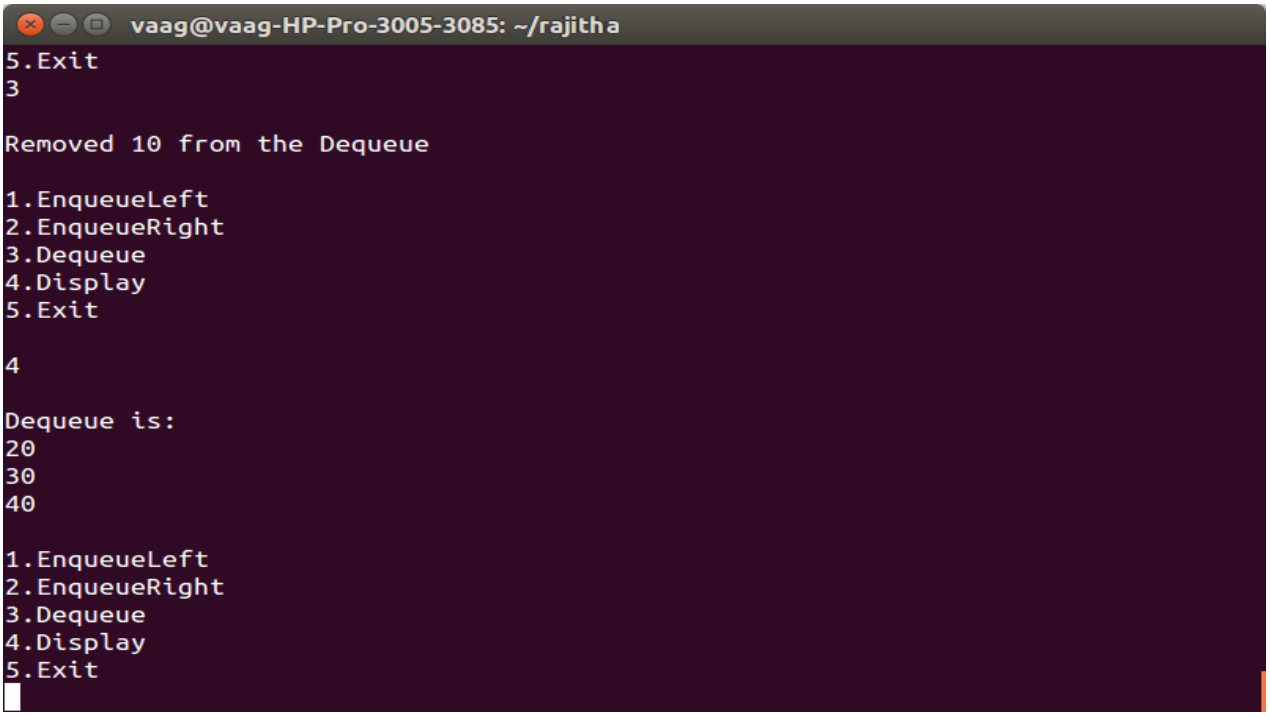

vaag@vaag-HP-Pro-3005-3085: ~/rajitha
Enter the Element: 40
1.Enqueue
2.DequeueRight
3.DequeueLeft
4.Display
5.Exit
4
Dequeue is:
10
20
30
40
1.Enqueue
2.DequeueRight
3.DequeueLeft
4.Display
5.Exit
2
Removed 40 from the Dequeue


vaag@vaag-HP-Pro-3005-3085: ~/rajitha
4.Display
5.Exit
3
Removed 10 form the Dequeue
1.Enqueue
2.DequeueRight
3.DequeueLeft
4.Display
5.Exit
4
Dequeue is:
20
30
1.Enqueue
2.DequeueRight
3.DequeueLeft
4.Display
5.Exit


vaag@vaag-HP-Pro-3005-3085: ~/rajitha
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ g++ deque.cpp
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ ./a.out
1.Input Restricted Deque
2.Output Restricted Deque
Enter your Choice: 2
1.EnqueueLeft
2.EnqueueRight
3.Dequeue
4.Display
5.Exit
1
Enter the Element: 10
1.EnqueueLeft
2.EnqueueRight
3.Dequeue
4.Display
5.Exit
2


vaag@vaag-HP-Pro-3005-3085: ~/rajitha
Enter the Element: 20
1.EnqueueLeft
2.EnqueueRight
3.Dequeue
4.Display
5.Exit
2
Enter the Element: 30
1.EnqueueLeft
2.EnqueueRight
3.Dequeue
4.Display
5.Exit
2
Enter the Element: 40
1.EnqueueLeft
2.EnqueueRight
3.Dequeue


vaag@vaag-HP-Pro-3005-3085: ~/rajitha
1.EnqueueLeft
2.EnqueueRight
3.Dequeue
4.Display
5.Exit
4
Dequeue is:
10
20
30
40
1.EnqueueLeft
2.EnqueueRight
3.Dequeue
4.Display
5.Exit
3
Removed 10 from the Dequeue


vaag@vaag-HP-Pro-3005-3085: ~/rajitha
5.Exit
3
Removed 10 from the Dequeue
1.EnqueueLeft
2.EnqueueRight
3.Dequeue
4.Display
5.Exit
4
Dequeue is:
20
30
40
1.EnqueueLeft
2.EnqueueRight
3.Dequeue
4.Display
5.Exit

6. Write a c++ program on double ended queue using double linked list.

```
#include<iostream>
#include<stdlib.h>
using namespace std;
class node
{
   public:
     int data;
     class  node *next;
     class node *prev;
};
class dqueue: public node
{
   node *head,*tail;
   int top1,top2;
   public:
      dqueue()
      {
        top1=0;
        top2=0;
        head=NULL;
        tail=NULL;
      }
      void push(int x)
      {
        node *temp;
        int ch;
        if(top1+top2 >=5)
        {
           cout <<"dqueue overflow";
           return ;
        }
        if(top1+top2 == 0)
        {
           head = new node;
           head->data=x;
           head->next=NULL;
           head->prev=NULL;
           tail=head;
           top1++;
        }
        else
        {
           cout <<" Add element 1.FIRST 2.LAST\n enter ur choice:";
           cin >> ch;
           if(ch==1)
           {
              top1++;
              temp=new node;
              temp->data=x;
              temp->next=head;
```

```
        temp->prev=NULL;
        head->prev=temp;
        head=temp;
         }
      else
      {
        top2++;
        temp=new node;
        temp->data=x;
        temp->next=NULL;
        temp->prev=tail;
        tail->next=temp;
        tail=temp;
      }
   }
}
void pop()
{
   int ch;
   cout <<"Delete 1.First Node 2.Last Node\n Enter ur choice:";
   cin >>ch;
   if(top1 + top2 <=0)
   {
      cout <<"\nDqueue under flow";
      return;
   }
   if(ch==1)
   {
      head=head->next;
      head->prev=NULL;
      top1--;
   }
   else
   {
      top2--;
      tail=tail->prev;
      tail->next=NULL;
   }
}
void display()
{
   int ch;
   node *temp;
   cout<<"display from 1.Staring 2.Ending\n Enter ur choice";
   cin>>ch;
   if(top1+top2<=0)
   {
      cout <<"under flow";
      return ;
   }
   if(ch==1)
```

```
        {
           temp=head;
           while(temp!=NULL)
           {
              cout<<temp->data<<" ";
              temp=temp->next;
           }
        }
        else
        {
           temp=tail;
           while(temp!=NULL)
           {
              cout<<temp->data<< " ";
              temp=temp->prev;
           }
        }
     }
};
int main()
{
   dqueue d1;
   int ch;
   while (1)
   {
      cout <<"\n1.INSERT  2.DELETE  3.DISPLAY  4.EXIT\n Enter ur choice:";
      cin >>ch;
      switch(ch)
      {
         case 1:cout<<"enter element";
              cin>>ch;
              d1.push(ch);
              break;
         case 2:d1.pop();
              break;
         case 3:d1.display();
              break;
         case 4:exit(1);
              break;
      }
   }
}
```

output:

```
vaag@vaag-HP-Pro-3005-3085: ~
vaag@vaag-HP-Pro-3005-3085:~$ g++ dque.cpp
vaag@vaag-HP-Pro-3005-3085:~$ ./a.out

1.INSERT  2.DELETE  3.DISPLAU  4.EXIT
 Enter ur choice:1
enter element11

1.INSERT  2.DELETE  3.DISPLAU  4.EXIT
 Enter ur choice:1
enter element12
 Add element 1.FIRST 2.LAST
 enter ur choice:2

1.INSERT  2.DELETE  3.DISPLAU  4.EXIT
 Enter ur choice:3
display from 1.Staring 2.Ending
 Enter ur choice1
11 12
1.INSERT  2.DELETE  3.DISPLAU  4.EXIT
 Enter ur choice:1
enter element10
```

```
vaag@vaag-HP-Pro-3005-3085: ~
1.INSERT  2.DELETE  3.DISPLAU  4.EXIT
 Enter ur choice:1
enter element10
 Add element 1.FIRST 2.LAST
 enter ur choice:1

1.INSERT  2.DELETE  3.DISPLAU  4.EXIT
 Enter ur choice:3
display from 1.Staring 2.Ending
 Enter ur choice2
12 11 10
1.INSERT  2.DELETE  3.DISPLAU  4.EXIT
 Enter ur choice:2
Delete 1.First Node 2.Last Node
 Enter ur choice:1

1.INSERT  2.DELETE  3.DISPLAU  4.EXIT
 Enter ur choice:3
display from 1.Staring 2.Ending
 Enter ur choice1
11 12
1.INSERT  2.DELETE  3.DISPLAU  4.EXIT
 Enter ur choice:4
vaag@vaag-HP-Pro-3005-3085:~$
```

7. Write C++ programs, using class templates, that use non-recursive functions to traverse the given binary tree in a) preorder b) inorder and c) postorder.

```
# include <iostream>
# include <stdlib.h>
using namespace std;
struct node
{
   int ele;
   node *left;
   node *right;
};
typedef struct node *nodeptr;
class stack
{
   private:
     struct snode
     {
        nodeptr ele;
        snode *next;
     };
     snode *top;
   public:
     stack()       //constructor
     {
        top=NULL;
     }
     void push(nodeptr p)
          {
             snode *temp;
             temp = new snode;
             temp->ele = p;
             temp->next = top;
             top=temp;
          }
          void pop()
          {
             if (top != NULL)
             {
                nodeptr t;
                     snode *temp;
                     temp = top;
                     top=temp->next;
                     delete temp;
             }
          }
          nodeptr topele()
          {
             if (top !=NULL)
                     return top->ele;
             else
                     return NULL;
          }
```

```
        int isempty()
        {
           return ((top == NULL) ? 1 : 0);
        }
};
class bstree
{
    public:
        void insert(int,nodeptr &);
        void del(int,nodeptr &);
        int deletemin(nodeptr &p);
        //void preorder(nodeptr);
        //void inorder(nodeptr);
        //void postorder(nodeptr);
        void preordernr(nodeptr);
        void inordernr(nodeptr);
        void postordernr(nodeptr);
};
void bstree::insert(int x,nodeptr &p)
{
    if(p==NULL)
    {
        p = new node;
        p->ele=x;
        p->left=NULL;
        p->right=NULL;
    }
    else
    {
        if (x < p->ele)
           insert(x,p->left);
        else if (x>p->ele)
           insert(x,p->right);
        else
           cout<<"Element already Exits !";
    }
}
void bstree:: del(int x,nodeptr &p)
{
    nodeptr d;
    if (p==NULL)
         cout<<"Element not found ";
    else if (x < p->ele)
         del(x,p->left);
    else if (x > p->ele)
         del(x,p->right);
    else if ((p->left == NULL) && (p->right ==NULL))
    {
       d=p;
         free(d);
         p=NULL;
    }
```

```
    else if (p->left == NULL)
    {
            d=p;
            free(d);
            p=p->right;
    }
    else if (p->right ==NULL)
    {
            d=p;
            p=p->left;
            free(d);
    }
    else
            p->ele=deletemin(p->right);
}
int bstree::deletemin(nodeptr &p)
{
    int c;
    if (p->left == NULL)
    {
            c=p->ele;
            p=p->right;
            return c;
    }
    else
            c=deletemin(p->left);
    return c;
}
/*void bstree::preorder(nodeptr p)
{
    if (p!=NULL)
    {
            cout<<p->ele<<"-->";
            preorder(p->left);
            preorder(p->right);
    }
}
void bstree::inorder(nodeptr p)
{
    if (p!=NULL)
    {
            inorder(p->left);
            cout<<p->ele<<"-->";
            inorder(p->right);
    }
}
void bstree::postorder(nodeptr p)
{
    if (p!=NULL)
    {
       postorder(p->left);
       postorder(p->right);
       cout<<p->ele<<"-->";
```

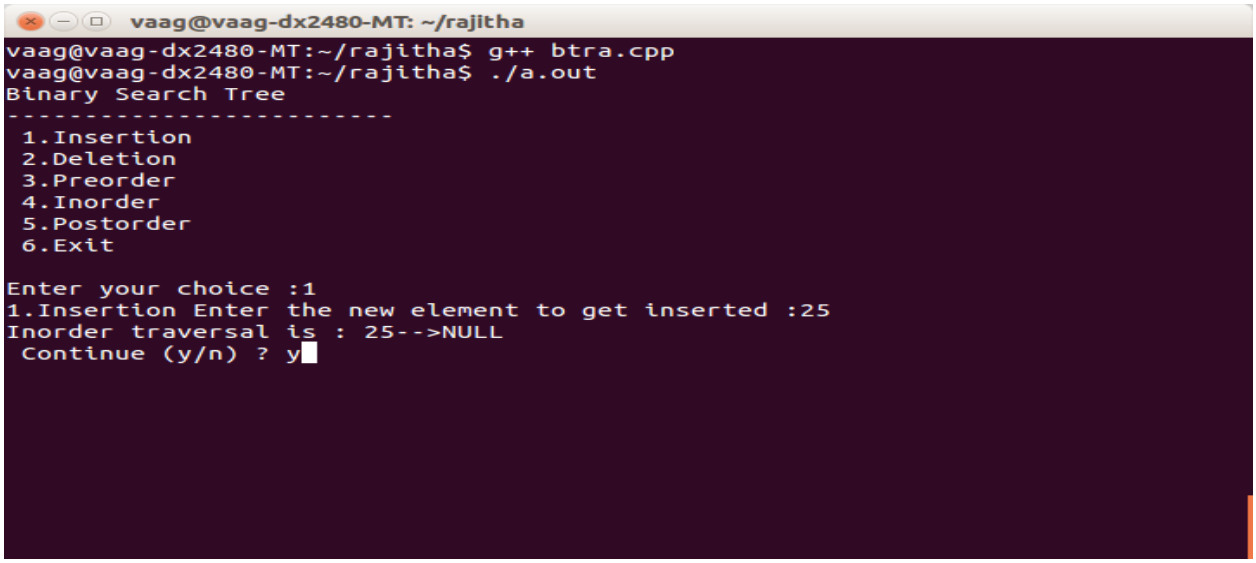
```
    }
}*/
void bstree::preordernr(nodeptr p)
{
    stack s;
    while (1)
    {
            if(p!=NULL)
            {
                cout<<p->ele<<"-->";
           s.push(p);
           p=p->left;
       }
            else if(s.isempty())
            {
           cout<<"Stack is empty";
           return;
       }
            else
            {
                nodeptr t;
                t=s.topele();
                p=t->right;
                s.pop();
            }
    }
}
void bstree::inordernr(nodeptr p)
{
    stack s;
    while (1)
    {
            if (p != NULL)
      {
                s.push(p);
                p=p->left;
            }
            else
            {
                if(s.isempty())
                {
             cout<<"NULL";
                        return;
          }
                else
                {
                        p=s.topele();
                        cout<<p->ele<<"-->";
                }
                s.pop();
                p=p->right;
            }
    }
```

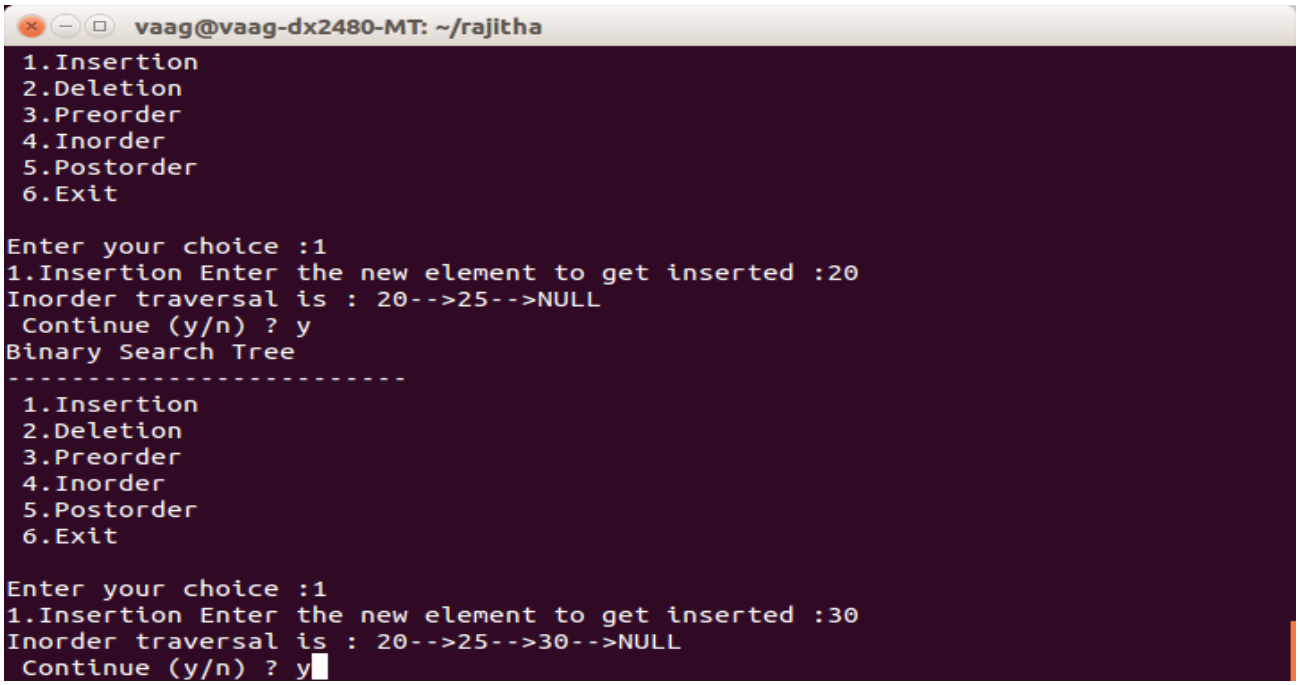
```
}
void bstree::postordernr(nodeptr p)
{
    stack s;
    while (1)
    {
       if(p!=NULL)
       {
          s.push(p);
          p=p->left;
       }
       else
       {
          if(s.isempty())
          {
             cout<<"Stack is empty";
             return;
          }
          else if(s.topele()->right==NULL)
          {
                        p=s.topele();
                        s.pop();
                        cout<<p->ele<<"-->";
                        if(p==s.topele()->right)
                        {
                           cout<<s.topele()->ele<<"-->";
                           s.pop();
                        }
                }
          if(!s.isempty())
                        p=s.topele()->right;
                else
                        p=NULL;
       }
    }
}
int main()
{
    int ch,x;
    bstree bst;
    char c='y';
    nodeptr root;
    root=NULL;
    do
    {
       //system("clear");
       cout<<"Binary Search Tree \n";
            cout<<"------------------------ \n ";
            cout<<"1.Insertion \n 2.Deletion \n ";
            cout<<"3.Preorder \n 4.Inorder \n 5.Postorder \n 6.Exit \n ";
            cout<<" \nEnter your choice :";
            cin>>ch;
            switch(ch)
```

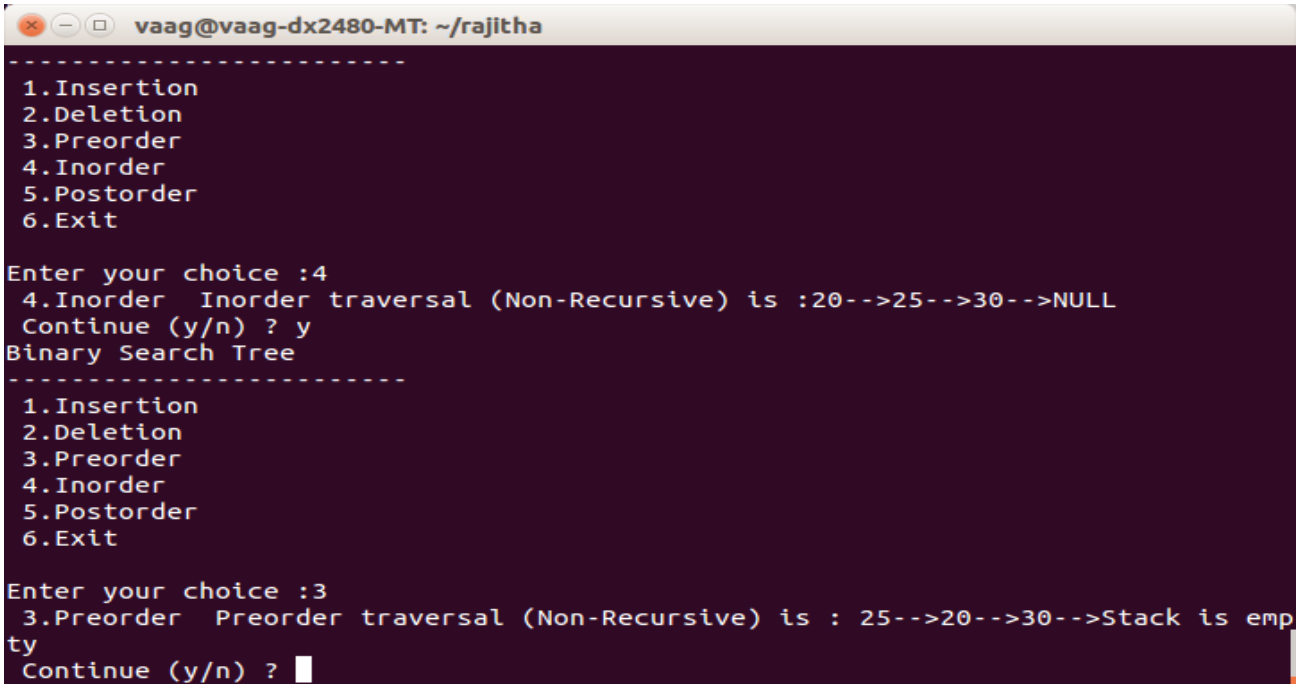
```
{
   case 1:cout<<"1.Insertion ";
            cout<<"Enter the new element to get inserted :";
            cin>>x;
            bst.insert(x,root);
            cout<<"Inorder traversal is : ";
            bst.inordernr(root);
            break;
   case 2:cout<<" 2.Deletion ";
            cout<<"Enter the element to get deleted : ";
            cin>>x;
            bst.del(x,root);
            bst.inordernr(root);
            break;
   case 3:cout<<" 3.Preorder ";
            if (root==NULL)
                  cout<<" Tree is empty";
            else
            {
                  cout<<" Preorder traversal (Non-Recursive) is : ";
                  bst.preordernr(root);
                  /*cout<<"Preorder traversal (Recursive) is : ";
                  bst.preorder(root);*/
            }
            break;
   case 4:cout<<" 4.Inorder ";
            if (root==NULL)
                  cout<<" Tree is empty";
            else
            {
                  cout<<" Inorder traversal (Non-Recursive) is :";
                  bst.inordernr(root);
                  /*cout<<" Inorder traversal (Recursive) is : ";
                  bst.inorder(root);*/
            }
            break;
   case 5:cout<<"5.Postorder ";
            if (root==NULL)
                  cout<<"Tree is empty";
            else
            {
                  cout<<"Postorder traversal (Non-Recursive) is : ";
                  bst.postordernr(root);
                  /*cout<<"Postorder traversal (Recursive) is : ";
                  bst.postorder(root);*/
            }
            break;
   case 6:exit(0);
}
```

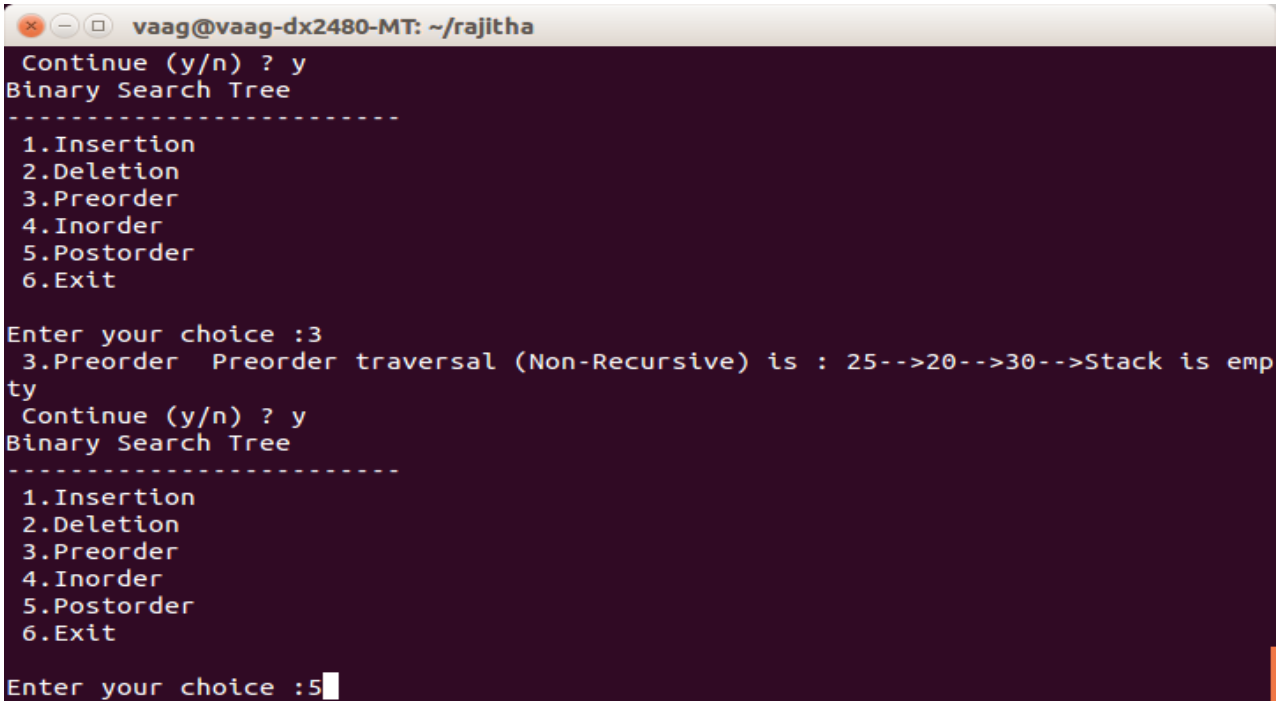
```
        cout<<" \n Continue (y/n) ? ";
          cin>>c;
    }while (c=='y' || c == 'Y');
    return 0;
}
```

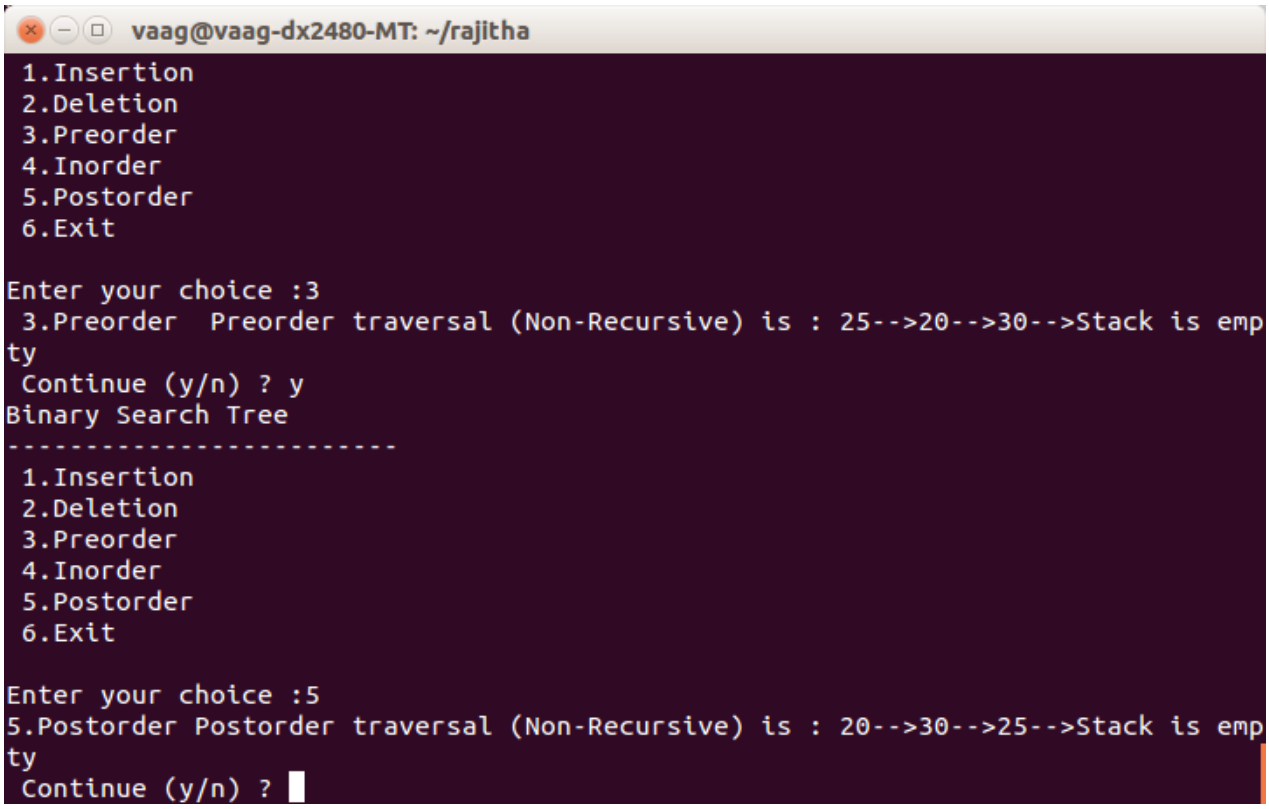
output:




vaag@vaag-dx2480-MT: ~/rajitha
1.Insertion
2.Deletion
3.Preorder
4.Inorder
5.Postorder
6.Exit
Enter your choice :1
1.Insertion Enter the new element to get inserted :20
Inorder traversal is : 20-->25-->NULL
Continue (y/n) ? y
Binary Search Tree
1.Insertion
2.Deletion
3.Preorder
4.Inorder
5.Postorder
6.Exit
Enter your choice :1
1.Insertion Enter the new element to get inserted :30
Inorder traversal is : 20-->25-->30-->NULL
Continue (y/n) ? y


vaag@vaag-dx2480-MT: ~/rajitha
1.Insertion
2.Deletion
3.Preorder
4.Inorder
5.Postorder
6.Exit
Enter your choice :4
4.Inorder Inorder traversal (Non-Recursive) is :20-->25-->30-->NULL
Continue (y/n) ? y
Binary Search Tree
1.Insertion
2.Deletion
3.Preorder
4.Inorder
5.Postorder
6.Exit
Enter your choice :3
3.Preorder Preorder traversal (Non-Recursive) is : 25-->20-->30-->Stack is empty
Continue (y/n) ?


vaag@vaag-dx2480-MT: ~/rajitha
Continue (y/n) ? y
Binary Search Tree
1.Insertion
2.Deletion
3.Preorder
4.Inorder
5.Postorder
6.Exit
Enter your choice :3
3.Preorder Preorder traversal (Non-Recursive) is : 25-->20-->30-->Stack is empty
Continue (y/n) ? y
Binary Search Tree
1.Insertion
2.Deletion
3.Preorder
4.Inorder
5.Postorder
6.Exit
Enter your choice :5

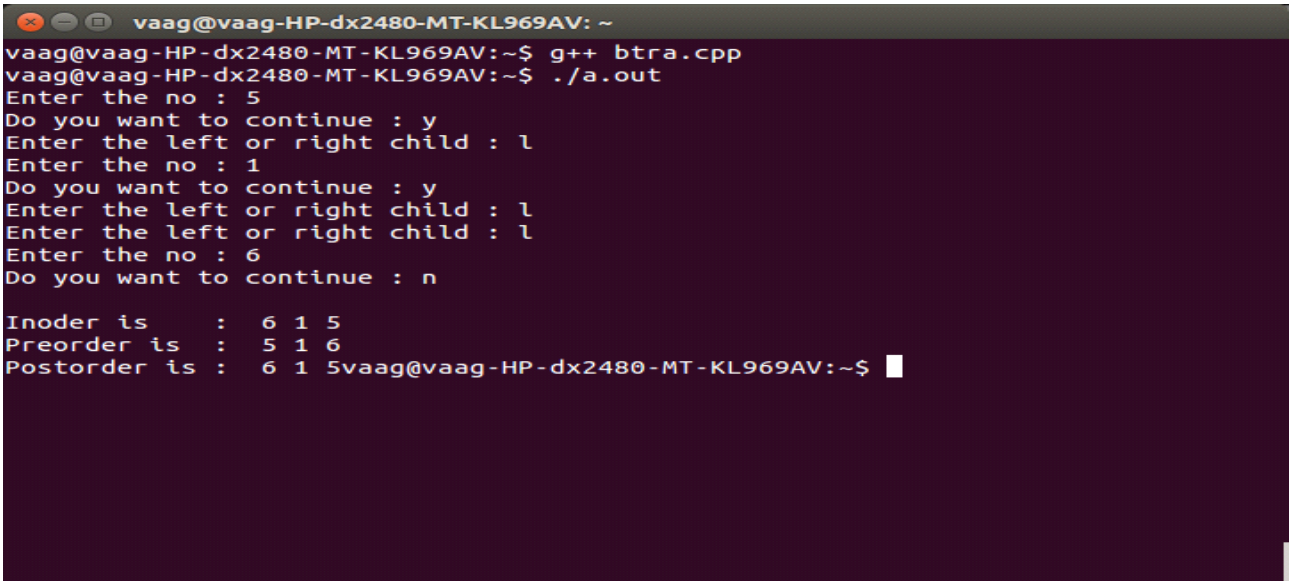

vaag@vaag-dx2480-MT: ~/rajitha
1.Insertion
2.Deletion
3.Preorder
4.Inorder
5.Postorder
6.Exit
Enter your choice :3
3.Preorder Preorder traversal (Non-Recursive) is : 25-->20-->30-->Stack is empty
Continue (y/n) ? y
Binary Search Tree
1.Insertion
2.Deletion
3.Preorder
4.Inorder
5.Postorder
6.Exit
Enter your choice :5
5.Postorder Postorder traversal (Non-Recursive) is : 20-->30-->25-->Stack is empty
Continue (y/n) ?

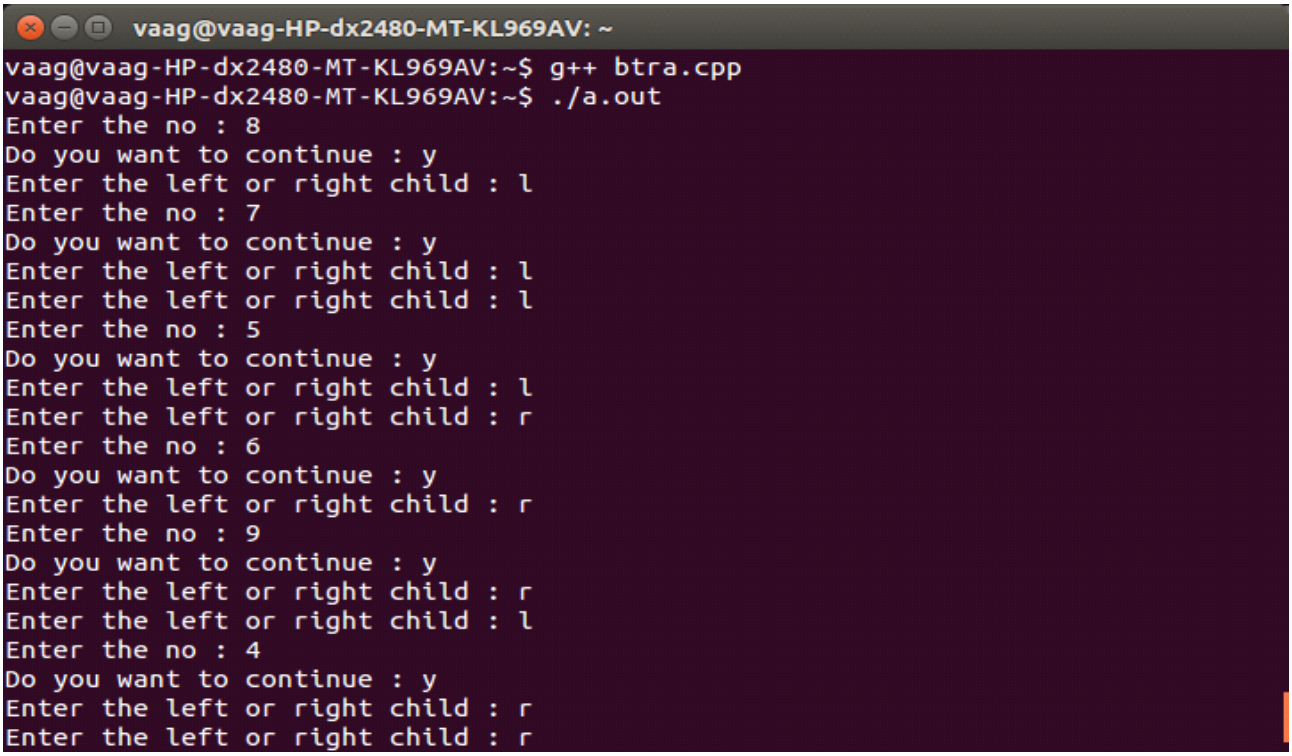
8. Write a c++ program using class templates, that use recursive functions to traverse the given binary tree in
a)preorder b)inorder c)postorder.

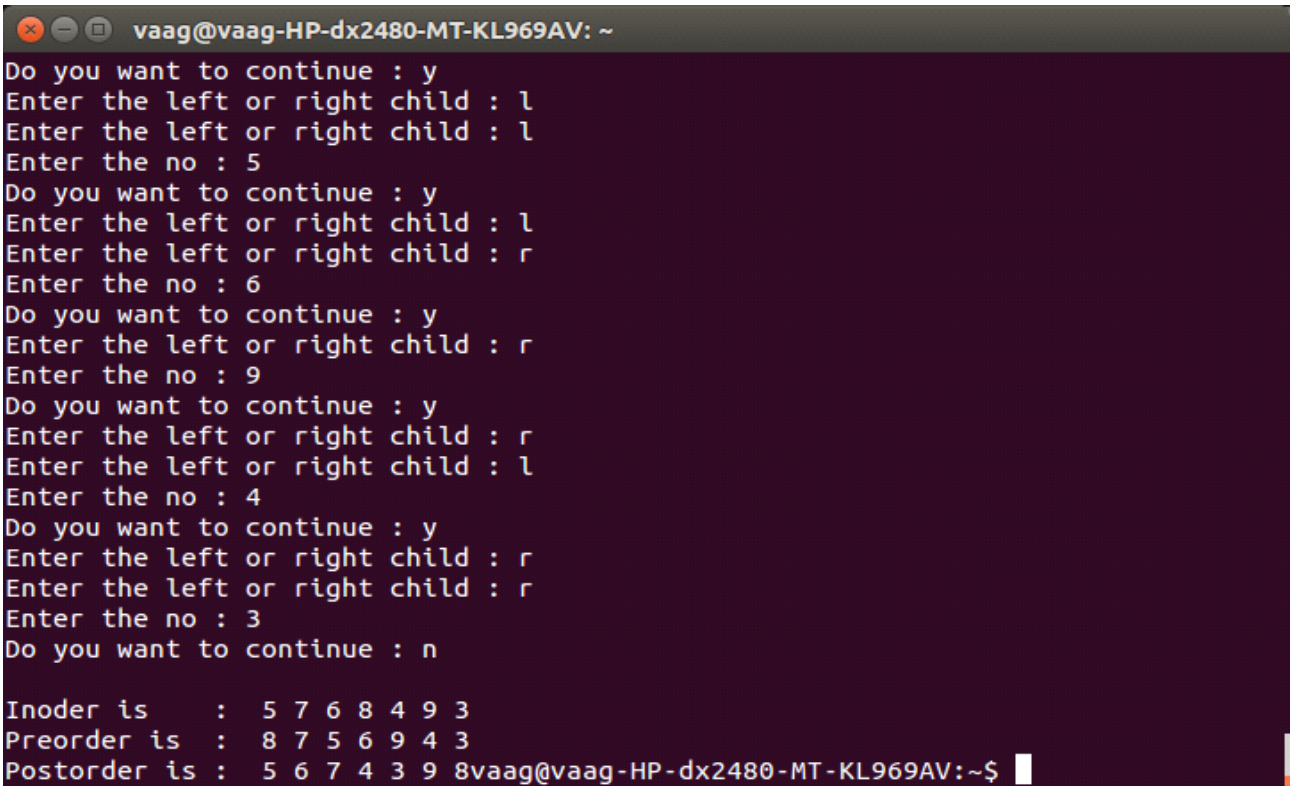
```
#include<iostream>
#include<stdio.h>
using namespace std;
struct btree
{
   struct btree *left;
   struct btree *right;
   int no;
};
void postorder(struct btree *trav);
void inorder(struct btree *trav);
void preorder(struct btree *trav);
struct btree * create(struct btree *trav);
int main()
{
   struct btree *root=NULL;
   char c;
   while(1)
   {
          root=create(root);
          cout<<"Do you want to continue : ";
          cin>>c;
          if(c=='n' ||c=='N')
             break;
   }
   cout<<endl<<"Inoder is    : ";
   inorder(root);
   cout<<endl<<"Preorder is  : ";
   preorder(root);
   cout<<endl<<"Postorder is : ";
   postorder(root);
}
struct btree * create(struct btree *trav)
{
   if(trav==NULL)
   {
          trav=new btree;
          trav->right=NULL;
          trav->left=NULL;
          cout<<"Enter the no : ";
          cin>>trav->no;
          return(trav);
   }
   char choice;
   cout<<"Enter the left or right child : ";
   cin>>choice;
   if(choice == 'r' || choice == 'R')
   {
```

```
        trav->right=create(trav->right);
    }
    if(choice=='l' || choice=='L')
    {
        trav->left=create(trav->left);
    }
    return(trav);
}
void inorder(struct btree *trav)
{
    if(trav==NULL)
        return ;
    inorder(trav->left);
    cout<<" "<<trav->no;
    inorder(trav->right);
}
void preorder(struct btree *trav)
{
    if(trav==NULL)
        return;
    cout<<" "<<trav->no;
    preorder(trav->left);
    preorder(trav->right);
}
void postorder(struct btree *trav)
{
    if(trav==NULL)
        return;
    postorder(trav->left);
    postorder(trav->right);
    cout<<" "<<trav->no;
}
```

output:




vaag@vaag-HP-dx2480-MT-KL969AV: ~
vaag@vaag-HP-dx2480-MT-KL969AV:~$ g++ btra.cpp
vaag@vaag-HP-dx2480-MT-KL969AV:~$ ./a.out
Enter the no : 8
Do you want to continue : y
Enter the left or right child : l
Enter the no : 7
Do you want to continue : y
Enter the left or right child : l
Enter the left or right child : l
Enter the no : 5
Do you want to continue : y
Enter the left or right child : l
Enter the left or right child : r
Enter the no : 6
Do you want to continue : y
Enter the left or right child : r
Enter the no : 9
Do you want to continue : y
Enter the left or right child : r
Enter the left or right child : l
Enter the no : 4
Do you want to continue : y
Enter the left or right child : r
Enter the left or right child : r


vaag@vaag-HP-dx2480-MT-KL969AV: ~
Do you want to continue : y
Enter the left or right child : l
Enter the left or right child : l
Enter the no : 5
Do you want to continue : y
Enter the left or right child : l
Enter the left or right child : r
Enter the no : 6
Do you want to continue : y
Enter the left or right child : r
Enter the no : 9
Do you want to continue : y
Enter the left or right child : r
Enter the left or right child : l
Enter the no : 4
Do you want to continue : y
Enter the left or right child : r
Enter the left or right child : r
Enter the no : 3
Do you want to continue : n
Inoder is : 5 7 6 8 4 9 3
Preorder is : 8 7 5 6 9 4 3
Postorder is : 5 6 7 4 3 9 8vaag@vaag-HP-dx2480-MT-KL969AV:~$

9. Write a C++ program using class templates to perform the following operations:
a) Insert an element into a binary search tree.
b) Delete an element from a binary search tree.
c) Search for a key element in a binary search tree.

```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct node
{
    int ele;
    node *left;
    node *right;
};
typedef struct node *nodeptr;
template<class t>
class bstree
{
    public:
       void insert(t, nodeptr &p);
       void del(t x, nodeptr &p);
       void find(t x, nodeptr &p);
       int deletemin(nodeptr &p);
};
template<class t>
void bstree<t>::insert(t x, nodeptr &p)
{
    if(p==NULL)
    {
        p=new node;
        p->ele=x;
        p->left=NULL;
        p->right=NULL;
    }
    else
    {
        if(x<p->ele)
            insert(x,p->left);
        else if(x>p->ele)
            insert(x,p->right);
        else
            cout<<"ele already exist";
    }
}
template<class t>
void bstree<t>::del(t x,nodeptr &p)
{
    nodeptr d;
    if(p==NULL)
        cout<<"ele not found";
    else if(x<p->ele)
```

```
        del(x,p->left);
    else if(x>p->ele)
        del(x,p->right);
    else if((p->left==NULL)&&(p->right==NULL))
    {
        d=p;
        free(d);
        cout<<"\n ele is deleted\n";
        p=NULL;
    }
    else if(p->left==NULL)
    {
        d=p;
        free(d);
        cout<<"\n ele is deleted";
        p=p->right;
    }
    else if(p->right==NULL)
    {
        d=p;
        free(d);
        cout<<"ele is deleted";
        p=p->left;
    }
    else
        p->ele=deletemin(p->right);
}
template<class t>
void bstree<t>::find(t x, nodeptr &p)
{
    if(p==NULL)
        cout<<"ele not found";
    else
    {
        if(x<p->ele)
            find(x,p->left);
        else if(x>p->ele)
            find(x,p->right);
        else
            cout<<"ele found";
    }
}
template<class t>
int bstree<t>::deletemin(nodeptr &p)
{
    int c;
    if(p->left==NULL)
    {
        c=p->ele;
        p=p->right;
        return c;
```

```
    }
    else
        c=deletemin(p->left);
    return c;
}
int main()
{
    int ch,x;
    nodeptr root=NULL;
    bstree<int>bs1;
    while(1)
    {
        cout<<"\n1.insert\n2.delete\n3.find\n4.exit:";
        cout<<"\nenter ur choice:";
        cin>>ch;
        switch(ch)
        {
              case 1:cout<<"enter the ele:";
                cin>>x;
                bs1.insert(x,root);
                        break;
               case 2:cout<<"enter ele:";
                        cin>>x;
                        bs1.del(x,root);
                        break;
               case 3:cout<<"enter ele to be search:";
                        cin>>x;
                        bs1.find(x,root);
                        break;
               case 4:exit(0);
        }
    }
}
```

output:

```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ clear

vaag@vaag-HP-Pro-3005-3085:~/rajitha$ g++ bstt.cpp
vaag@vaag-HP-Pro-3005-3085:~/rajitha$ ./a.out

1.insert
2.delete
3.find
4.exit:
enter ur choice:1
enter the ele:11

1.insert
2.delete
3.find
4.exit:
enter ur choice:1
enter the ele:12

1.insert
2.delete
3.find
4.exit:
enter ur choice:1
```

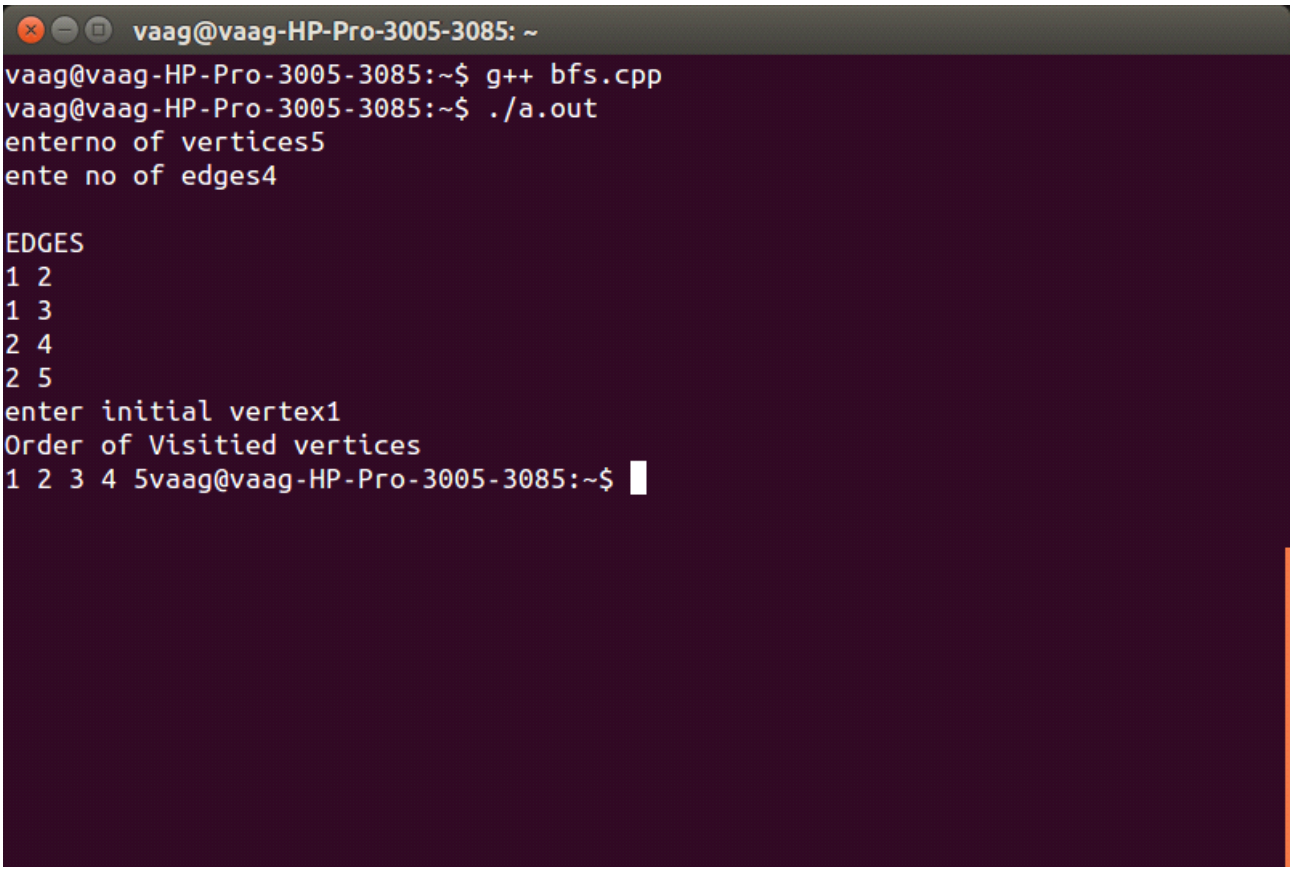
```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha

1.insert
2.delete
3.find
4.exit:
enter ur choice:1
enter the ele:13

1.insert
2.delete
3.find
4.exit:
enter ur choice:3
enter ele to be search:11
ele found
1.insert
2.delete
3.find
4.exit:
enter ur choice:3
enter ele to be search:15
ele not found
1.insert
2.delete
```

```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
1.insert
2.delete
3.find
4.exit:
enter ur choice:3
enter ele to be search:15
ele not found
1.insert
2.delete
3.find
4.exit:
enter ur choice:2
enter ele:11

 ele is deleted
1.insert
2.delete
3.find
4.exit:
enter ur choice:1
enter the ele:12
ele already exist
1.insert
2.delete
```

10. Write C++ programs using class templates for the implementation of bfs for a given graph.

```
#include<iostream>
#include<stdlib.h>
using namespace std;
int cost[10][10],i,j,k,n,queue[10],front,rear,v,visit[10],visited[10];
int main()
{
        int m;
        cout <<"enterno of vertices";
        cin >> n;
        cout <<"ente no of edges";
        cin >> m;
        cout <<"\nEDGES \n";
        for(k=1;k<=m;k++)
        {
                cin >>i>>j;
                cost[i][j]=1;
        }
        cout <<"enter initial vertex";
        cin >>v;
        cout <<"Order of Visitied vertices\n";
        cout << v;
        visited[v]=1;
        k=1;
        while(k<n)
        {
                for(j=1;j<=n;j++)
                        if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
                        {
                                visit[j]=1;
                                queue[rear++]=j;
                        }
                v=queue[front++];
                cout<< " "<<v;
                k++;
                visit[v]=0;
                visited[v]=1;
        }

}
```

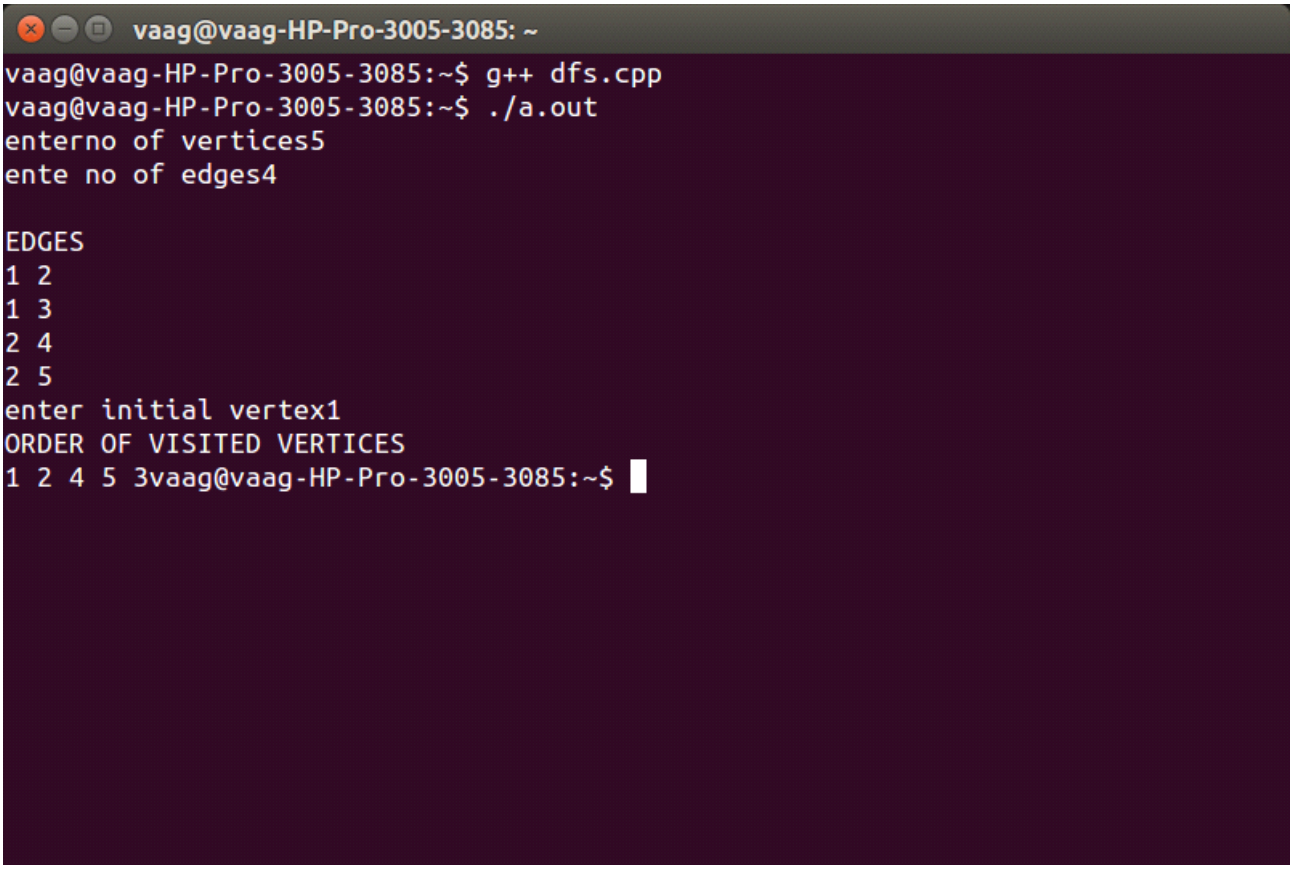
output:


vaag@vaag-HP-Pro-3005-3085: ~
vaag@vaag-HP-Pro-3005-3085:~$ g++ bfs.cpp
vaag@vaag-HP-Pro-3005-3085:~$ ./a.out
enterno of vertices5
ente no of edges4
EDGES
1 2
1 3
2 4
2 5
enter initial vertex1
Order of Visitied vertices
1 2 3 4 5vaag@vaag-HP-Pro-3005-3085:~$

10. Write C++ programs using class templates for the implementation of dfs for a given graph.

```
#include<iostream>
#include<stdlib.h>
using namespace std;
int cost[10][10],i,j,k,n,stack[10],top,v,visit[10],visited[10];
int main()
{
        int m;
        cout <<"enterno of vertices";
        cin >> n;
        cout <<"ente no of edges";
        cin >> m;
        cout <<"\nEDGES \n";
        for(k=1;k<=m;k++)
        {
                cin >>i>>j;
                cost[i][j]=1;
        }
        cout <<"enter initial vertex";
        cin >>v;
        cout <<"ORDER OF VISITED VERTICES\n";
        cout << v <<" ";
        visited[v]=1;
        k=1;
        while(k<n)
        {
                for(j=n;j>=1;j--)
                        if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
                        {
                                visit[j]=1;
                                stack [top]=j;
                                top++;
                        }
                v= stack [--top];
                cout<< " "<<v;
                k++;
                visit[v]=0; visited[v]=1;
        }
}
```

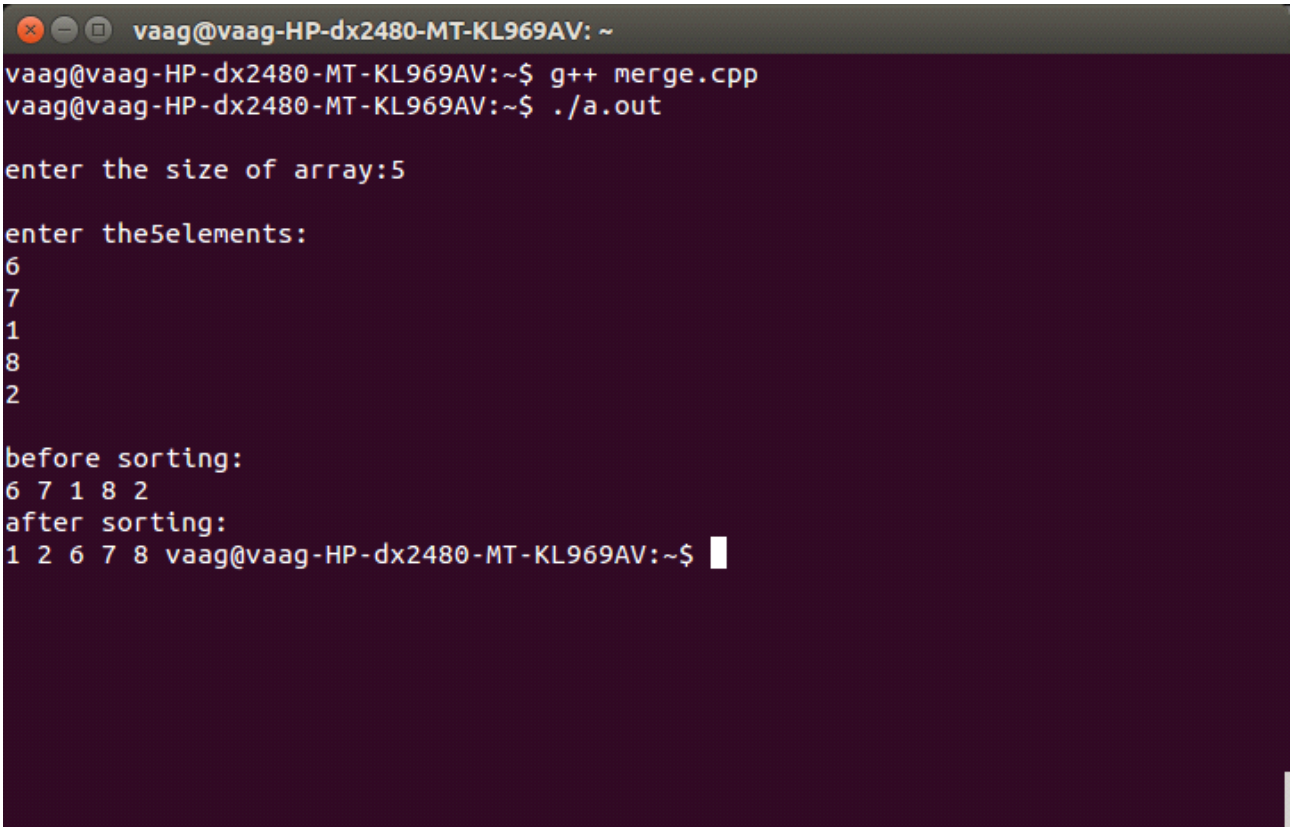
output:


vaag@vaag-HP-Pro-3005-3085: ~
vaag@vaag-HP-Pro-3005-3085:~$ g++ dfs.cpp
vaag@vaag-HP-Pro-3005-3085:~$ ./a.out
enterno of vertices5
ente no of edges4
EDGES
1 2
1 3
2 4
2 5
enter initial vertex1
ORDER OF VISITED VERTICES
1 2 4 5 3vaag@vaag-HP-Pro-3005-3085:~$

11. Write a c++ program using class template for the implementing the merge sort

```
#include<iostream>
#include<stdlib.h>
using namespace std;
void mergesort(int a[],int lb,int mid ,int ub)
{
    int i=lb,j=mid+1,k=0,b[50];
    while(i<=mid && j<=ub)
    {
       if(a[i]<=a[j])
          b[k++]=a[i++];
       else
          b[k++]=a[j++];
    }
    while(i<=mid)
       b[k++]=a[i++];
    while(j<=ub)
       b[k++]=a[j++];
    for(k=0;k<=ub-lb;k++)
       a[k+lb]=b[k];
}
void merge(int a[],int low,int high)
{
    int mid;
    if(low<high)
    {
       mid=(low+high)/2;
       merge(a,low,mid);
       merge(a,mid+1,high);
       mergesort(a,low,mid,high);
    }
}

int main()
{
    int i,a[30],n;
    cout<<"\nenter the size of array:";
    cin>>n;
    cout<<"\nenter the"<< n <<"elements:\n";
    for(i=0;i<n;i++)
       cin>>a[i];
    cout<<"\nbefore sorting:\n";
    for(i=0;i<n;i++)
       cout<<a[i]<<" ";
    merge(a,0,n-1);
    cout<<"\nafter sorting:\n";
    for(i=0;i<n;i++)
       cout<<a[i]<<" ";
}
```

output:


vaag@vaag-HP-dx2480-MT-KL969AV: ~
vaag@vaag-HP-dx2480-MT-KL969AV:~$ g++ merge.cpp
vaag@vaag-HP-dx2480-MT-KL969AV:~$ ./a.out
enter the size of array:5
enter the5elements:
6
7
1
8
2
before sorting:
6 7 1 8 2
after sorting:
1 2 6 7 8 vaag@vaag-HP-dx2480-MT-KL969AV:~$

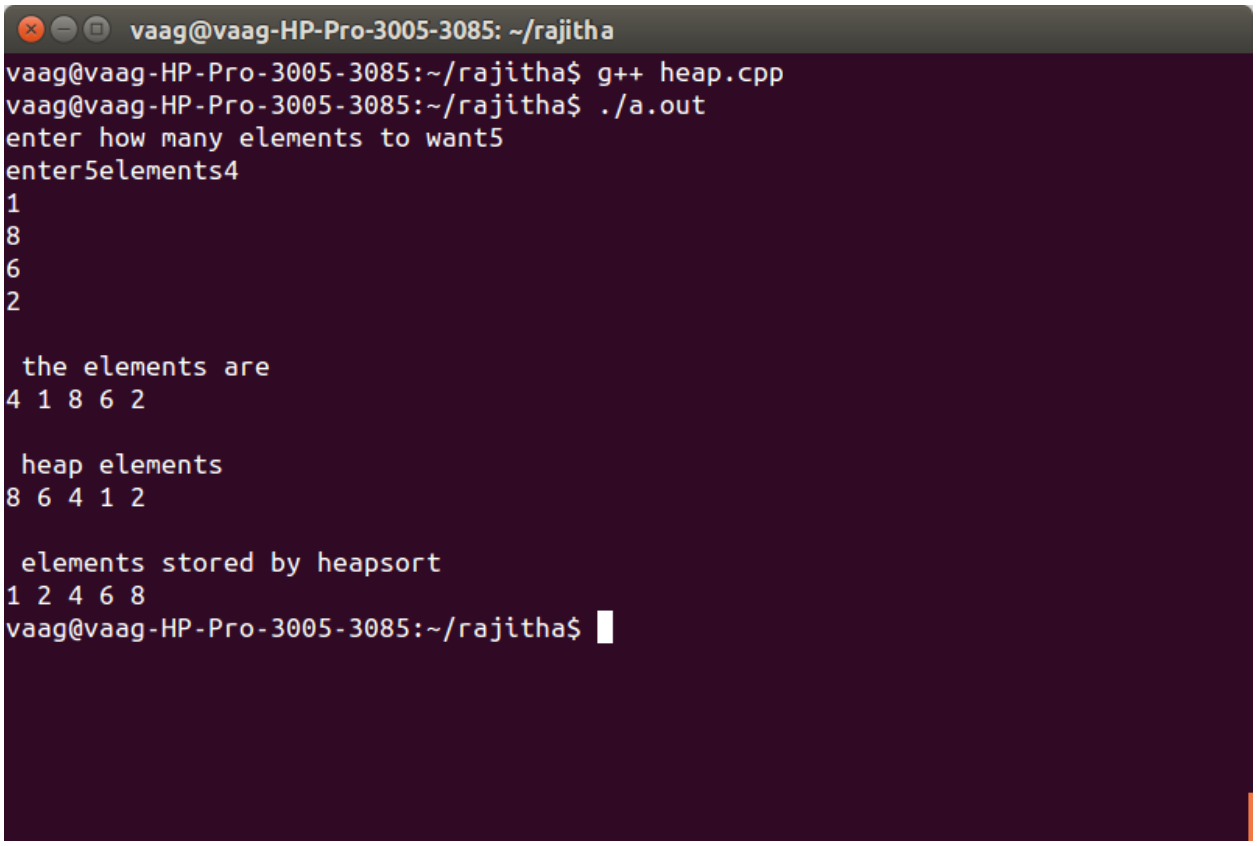
11. Write a c++ program using class template for the implementing the heap sort

```
#include<iostream>
#include<stdlib.h>
#define MAX 10
using namespace std;
template<class T>
class heap
{
    private:
      T arr[MAX];
           int n;
    public:
           heap();
           void insert(T num);
           void makeheap();
           void heapsort();
           void display();
};
template<class T>
heap<T>::heap()
{
    n=0;
    for(int i=0;i<MAX;i++)
      arr[i]=0;
}
template<class T>
void heap<T>::insert(T num)
{
    if(n<MAX)
    {
           arr[n]=num;
           n++;
    }
    else
      cout<<"array is full \n";
}
template<class T>
void heap<T>::makeheap()
{
    for(int i=1;i<n;i++)
    {
           T val=arr[i];
           int j=i;
           int f=(j-1)/2;
           while((j>0)&&(arr[f]<val))
           {
              arr[j]=arr[f];
              j=f;
         f=(j-1)/2;
           }
```

```
            arr[j]=val;
    }
}
template<class T>
void heap<T>::heapsort()
{
    for(int i=n-1;i>0;i--)
    {
            T temp=arr[i];
            arr[i]=arr[0];
            int k=0;
            int j;
            if(i==1)
               j=-1;
            else
               j=1;
            if(i>2&&arr[2]>arr[1])
           j=2;
            while(j>=0&&temp<arr[j])
            {
               arr[k]=arr[j];
               k=j;
               j=2*k+1;
               if(j+1<=j-1&&arr[j]<arr[j+1])
                        j++;
               if(j>i-1)
                        j=-1;
            }
            arr[k]=temp;
    }
}
template<class T>
void heap<T>::display()
{
    for(int i=0;i<n;i++)
            cout<<arr[i]<<" ";
    cout<<"\n";
}
int main()
{
    heap<int>obj;
    int i,n,x;
    cout<<"enter how many elements to want";
    cin>>n;
    cout<<"enter"<<n<<"elements";
    for(i=0;i<n;i++)
    {
            cin>>x;
            obj.insert(x);
    }
    cout<<"\n the elements are"<<endl;
```

```
    obj.display();
    obj.makeheap();
    cout<<"\n heap elements"<<endl;
    obj.display();
    obj.heapsort();
    cout<<"\n elements stored by heapsort"<<endl;
    obj.display();
}
```

output:



12. Write a C++ program using class templates to perform the following operations
a) Insertion into a B-tree b) Deletion from a B-tree

```
#include<iostream>
#include<stdlib.h>
using namespace std;
#define max 50
struct btree
{
    int n;
    int keys[max-1];
    struct btree* p[max];
}*root=NULL;
typedef struct btree node1;
int count=0,ele;
enum keystatus{duplicate,searchnodefailure,success,insertit,lesskeys};
enum keystatus ins(node1*r,int x,int *y,node1 **u);
enum keystatus del(node1 *r,int x);
void insertnode()
{
    node1* newnode;
    int upkey;
    enum keystatus value;
    cout<<"enter element to insert:";
    cin>>ele;
    value=ins(root,ele,&upkey,&newnode);
    if(value==duplicate)
    {
            cout<<"element already exist\n";
            return;
    }
    if(value==insertit)
    {
            node1*uproot=root;
            root=(node1*)malloc(sizeof(node1));
            root->n=1;
            root->keys[0]=upkey;
            root->p[0]=uproot;
            root->p[1]=newnode;
    }
    count++;
    cout<<"element is successfully inserted";
}
int searchnodepos(int key,int *key_arr,int n)
{
    int pos=0;
    while(pos<n && key>key_arr[pos])
       pos++;
    return pos;
}
```

```
enum keystatus ins(node1 *ptr,int key,int *upkey,node1 **newnode)
{
    node1 *newptr,*lastptr;
    int pos,i,n,splitpos;
    int newkey,lastkey;
    enum keystatus value;
    if(ptr==NULL)
    {
            *newnode=NULL;
            *upkey=key;
            return insertit;
    }
    n=ptr->n;
    pos=searchnodepos(key,ptr->keys,n);
    if(pos<n && key==ptr->keys[pos])
            return duplicate;
    value=ins(ptr->p[pos],key,&newkey,&newptr);
    if(value!=insertit)
            return value;
    if(n<max-1)
    {
            pos=searchnodepos(newkey,ptr->keys,n);
            for(i=n;i>pos;i--)
            {
               ptr->keys[i]=ptr->keys[i-1];
               ptr->p[i+1]=ptr->p[i];
            }
            ptr->keys[pos]=newkey;
            ptr->p[pos+1]=newptr;
            ++ptr->n;
            return success;
    }
    if(pos==max-1)
    {
            lastkey=newkey;
            lastptr=newptr;
    }
    else
    {
            lastkey=ptr->keys[max-2];
            lastptr=ptr->p[max-1];
            for(i=max-2;i>pos;i--)
            {
               ptr->keys[i]=ptr->keys[i-1];
               ptr->p[i+1]=ptr->p[i];
            }
            ptr->keys[pos]=newkey;
            ptr->p[pos+1]=newptr;
    }
    splitpos=(max-1)/2;
    (*upkey)=ptr->keys[splitpos];
```

```
    (*newnode)=(node1*)malloc(sizeof(node1));
    ptr->n=splitpos;
    (*newnode)->n=max-1-splitpos;
    for(i=0;i<(*newnode)->n;i++)
    {
          (*newnode)->p[i]=ptr->p[i+splitpos+1];
          if(i<(*newnode)->n-1)
             (*newnode)->keys[i]=ptr->keys[i+splitpos+1];
          else

             (*newnode)->keys[i]=lastkey;
    }
    (*newnode)->p[(*newnode)->n]=lastptr;
    return insertit;
}
void display(node1 *ptr,int blanks)
{
    if(count==0)
    {
          cout<<"btree is empty";
          return;
    }
    if(ptr)
    {
          int i;
          for(i=0;i<ptr->n;i++)
              cout<<ptr->keys[i]<<" ";
          cout<<endl;
          for(i=0;i<=ptr->n;i++)
              display(ptr->p[i],blanks+10);
    }
}
void search()
{
    int pos,i,n;
    node1*ptr=root;
    if(count==0)
    {
          cout<<"btree empty";
          return;
    }
    cout<<"enter element to search node:";
    cin>>ele;
    while(ptr)
    {
          n=ptr->n;
          pos=searchnodepos(ele,ptr->keys,n);
          if(pos<n && ele==ptr->keys[pos])
          {
              cout<<"element"<<ele<<"is found";
                  return;
```

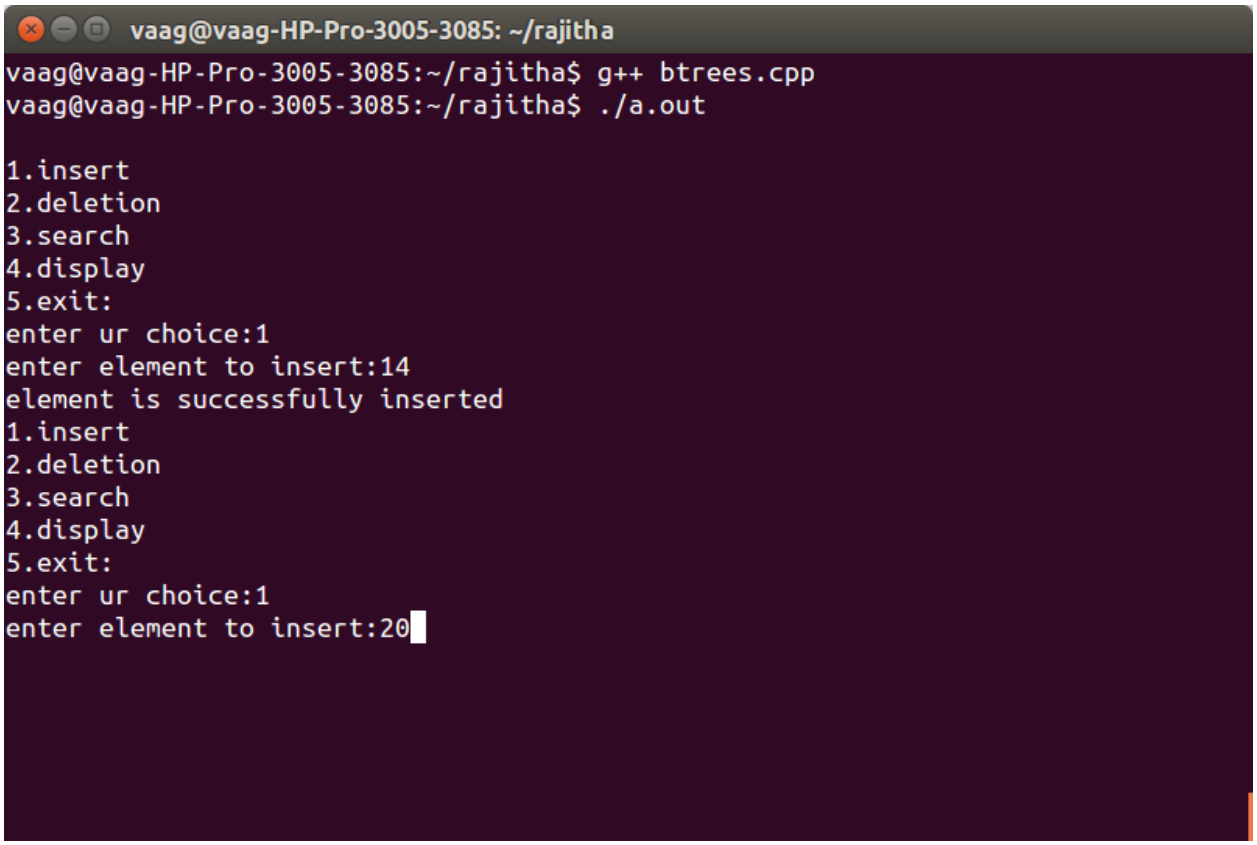
```
            }
        ptr=ptr->p[pos];
    }
    cout<<"element"<<ele<<"is not found";
}
void deletenode()
{
    int flag=1;
    node1* uproot;
    enum keystatus value;
    if(count==0)
    {
            cout<<"btree is empty";
            return;
    }
    cout<<"enter element to delete:";
    cin>>ele;
    value=del(root,ele);
    switch(value)
    {
            case searchnodefailure:cout<<"element"<<ele<<"is not available";
                flag=0;
                break;
            case lesskeys:uproot=root;
                root=root->p[0];
                free(uproot);
                break;
    }
    if(flag==1)
    {
            cout<<"element"<<ele<<"is deleted";
            count--;
    }
}
enum keystatus del(node1*ptr,int key)
{
    int pos,i,pivot,n,maxin;
    int *key_arr;
    enum keystatus value;
    node1**p,*lchild,*rchild;
    if(ptr==NULL)
            return searchnodefailure;
    n=ptr->n;
    key_arr=ptr->keys;
    p=ptr->p;
    maxin=(max-1)/2;
    pos=searchnodepos(key,key_arr,n);
    if(p[0]==NULL)
    {
            if(pos==n || key<key_arr[pos])
           return searchnodefailure;
```

```
   for(i=pos+1;i<n;i++)
   {
      key_arr[i-1]=key_arr[i];
      p[i]=p[i+1];
   }
   return --ptr->n>=(ptr==root ? 1:maxin) ? success:lesskeys;
}
if(pos<n && key==key_arr[pos])
{
   node1*qp=p[pos],*qp1;
   int nkey;
   while(1)
   {
      nkey=qp->n;
      qp1=qp->p[nkey];
      if(qp1==NULL)
          break;
      qp=qp1;
   }
   key_arr[pos]=qp->keys[nkey-1];
   qp->keys[nkey-1]=key;
 }
 value=del(p[pos],key);
 if(value!=lesskeys)
     return value;
 if(pos>0 && p[pos-1]->n>maxin)
 {
     pivot=pos-1;
     lchild=p[pivot];
     rchild=p[pos];
     rchild->p[rchild->n+1]=rchild->p[rchild->n];
     for(i=rchild->n;i>0;i--)
          {
                rchild->keys[i]=rchild->keys[i-1];
                rchild->p[i]=rchild->p[i-1];
          }
          rchild->n++;
          rchild->keys[0]=key_arr[pivot];
          rchild->p[0]=lchild->p[lchild->n];
          key_arr[pivot]=lchild->keys[--lchild->n];
          return success;
 }
 if(pos>maxin)
 {
          pivot=pos;
          lchild=p[pivot];
          rchild=p[pivot+1];
          lchild->keys[lchild->n]=key_arr[pivot];
          lchild->p[lchild->n+1]=rchild->p[0];
          key_arr[pivot]=rchild->keys[0];
          lchild->n++;
```

```
            rchild->n--;
        for(i=0;i<rchild->n;i++)
            {
                rchild->keys[i]=rchild->keys[i+1];
                rchild->p[i]=rchild->p[i+1];
            }
            rchild->p[rchild->n]=rchild->p[rchild->n+1];
            return success;
    }
    if(pos==n)
        pivot=pos-1;
    else
        pivot=pos;
    lchild=p[pivot];
    rchild=p[pivot+1];
    lchild->keys[lchild->n]=key_arr[pivot];
    lchild->p[lchild->n+1]=rchild->p[0];
    for(i=0;i<rchild->n;i++)
    {
        lchild->keys[lchild->n+1+i]=rchild->keys[i];
        lchild->p[lchild->n+2+i]=rchild->p[i+1];
    }
    lchild->n=lchild->n+rchild->n+1;
    free(rchild);
    for(i=pos+1;i<n;i++)
    {
        key_arr[i-1]=key_arr[i];
        p[i]=p[i+1];
    }
    return --ptr->n>=(ptr==root?1:maxin)?success:lesskeys;
}
int main()
{
    int op;
    while(1)
    {
        cout<<"\n1.insert\n2.deletion\n3.search\n4.display\n5.exit:\n";
       cout<<"enter ur choice:";
        cin>>op;
        switch(op)
        {
            case 1:insertnode();
                break;
            case 2:deletenode();
                break;
            case 3:search();
                break;
```

```
                case 4:cout<<"\nelements are:\n";
                       display(root,0);
                       break;
                case 5:exit(0);
                default:cout<<"invalid choice\n";
        }
    }
}
```

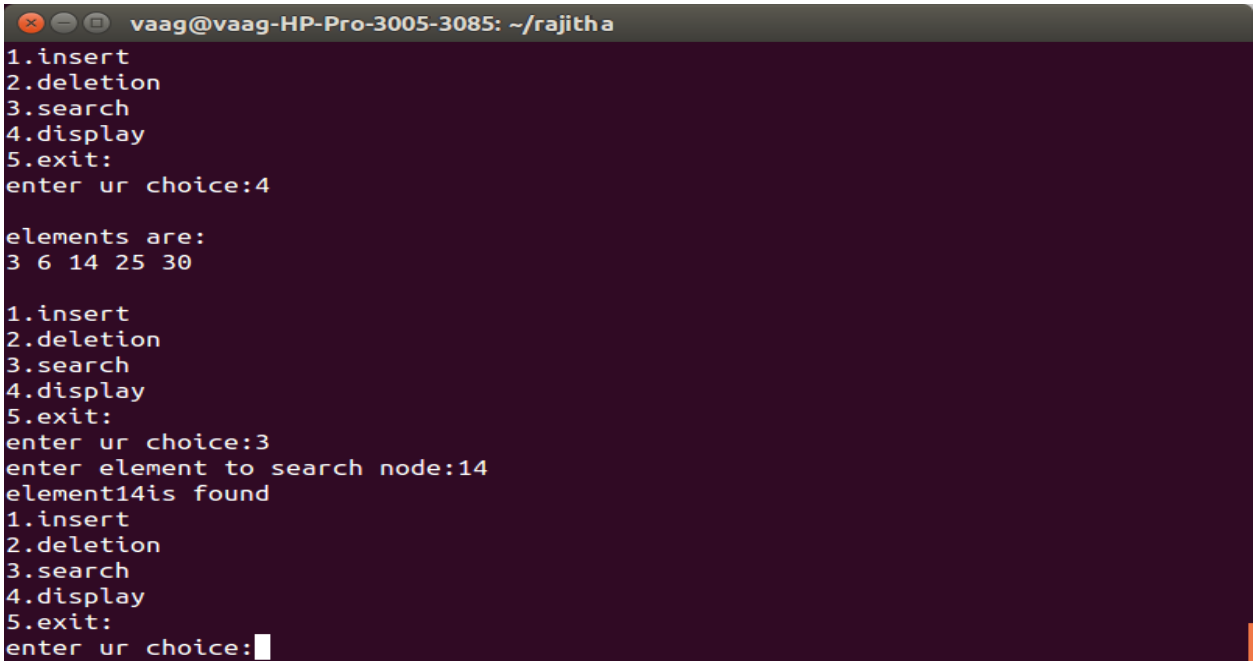
output:

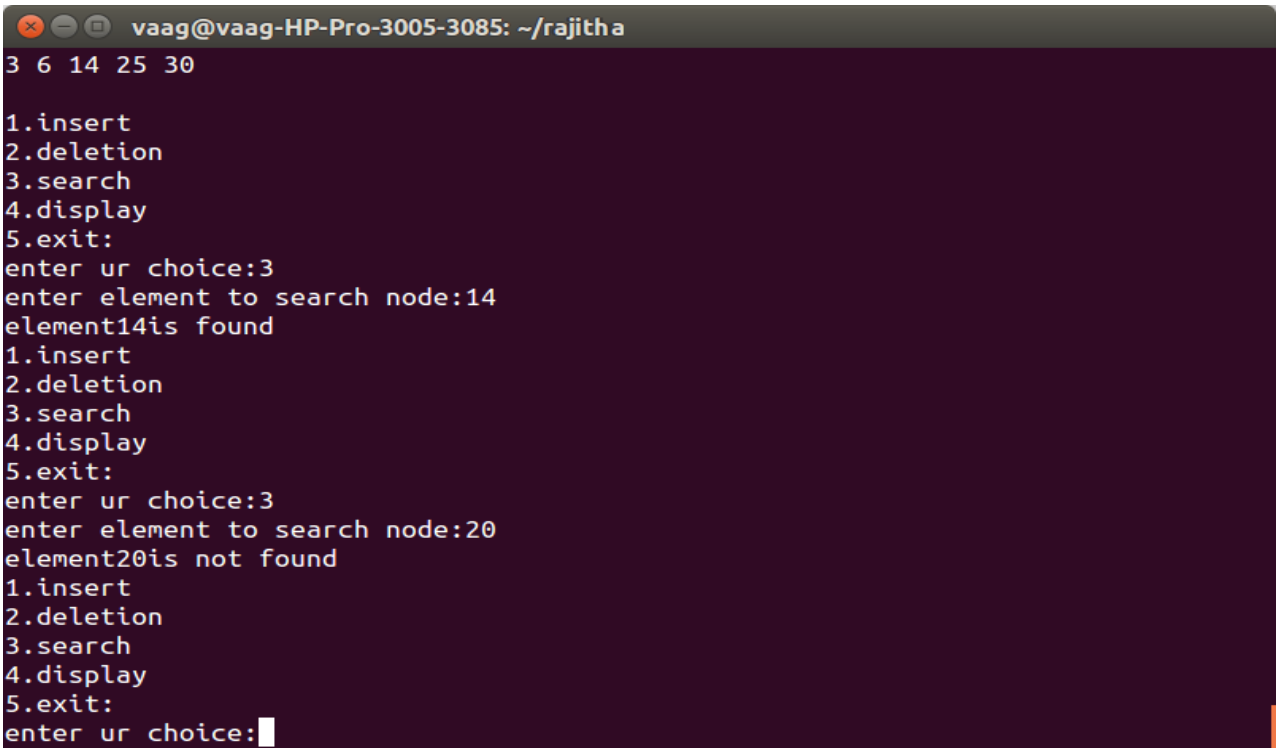


```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
enter element to insert:20
element is successfully inserted
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:1
enter element to insert:6
element is successfully inserted
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:1
enter element to insert:25
element is successfully inserted
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:
```

```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
enter element to insert:25
element is successfully inserted
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:1
enter element to insert:3
element is successfully inserted
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:1
enter element to insert:30
element is successfully inserted
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:
```

```
vaag@vaag-HP-Pro-3005-3085: ~/rajitha
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:4

elements are:
3 6 14 20 25 30

1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:2
enter element to delete:20
element20is deleted
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:
```


vaag@vaag-HP-Pro-3005-3085: ~/rajitha
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:4
elements are:
3 6 14 25 30
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:3
enter element to search node:14
element14is found
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:


vaag@vaag-HP-Pro-3005-3085: ~/rajitha
3 6 14 25 30
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:3
enter element to search node:14
element14is found
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:3
enter element to search node:20
element20is not found
1.insert
2.deletion
3.search
4.display
5.exit:
enter ur choice:

13. Write a C++ program using class templates to perform the following operations
1) insertion into an avl tree 2)deletion from an avl tree

```
#include<iostream>
#include<stdlib.h>
using namespace std;
#define TRUE 1
#define FALSE 0
//#define NULL 0
class AVL;
class AVLNODE
{
    friend class AVL;
    private:
        int data;
        AVLNODE *left,*right;
        int bf;
};
class AVL
{
    private:
        AVLNODE *root;
    public:
        AVLNODE *loc,*par;
        AVL()
        {
           root=NULL;
        }
        int insert(int);
        void removeitem(int);
        void remove1(AVLNODE *,AVLNODE *,int);
        void remove2(AVLNODE *,AVLNODE *,int);
        void search(int x);
        void search1(AVLNODE *,int);
};
int AVL::insert(int x)
{
    AVLNODE *a,*b,*c,*f,*p,*q,*y,*clchild,*crchild;
    int found,unbalanced;
    int d;
    if(!root)   //special case empty tree
    {
       y=new AVLNODE;
            y->data=x;
            root=y;
            root->bf=0;
            root->left=root->right=NULL;
            return TRUE;
    }
    f=NULL;
    a=p=root;
    q=NULL;
    found=FALSE;
    while(p&&!found)
    {
       //search for insertion point for x
            if(p->bf)
```

```
        {
           a=p;
           f=q;
        }
        if(x<p->data)    //take left branch
        {
           q=p;
           p=p->left;
        }
        else if(x>p->data)
        {
           q=p;
           p=p->right;
        }
        else
        {
           y=p;
           found=TRUE;
        }
}     //end while
if(!found)
{
        y=new AVLNODE;
        y->data=x;
        y->left=y->right=NULL;
        y->bf=0;
        if(x<q->data)    //insert as left child
           q->left=y;
        else
           q->right=y;    //insert as right child
        if(x>a->data)
        {
           p=a->right;
           b=p;
           d=-1;
        }
        else
        {
           p=a->left;
           b=p;
           d=1;
        }
        while(p!=y)
          if(x>p->data)   //height of  right increases by 1
          {
                p->bf=-1;
                p=p->right;
          }
          else          //height of left increases by 1
          {
                p->bf=1;
                p=p->left;
          }
          //is tree unbalanced
          unbalanced=TRUE;
          if(!(a->bf)||!(a->bf+d))
          {       //tree still balanced
```

```
        a->bf+=d;
        unbalanced=FALSE;
    }
    if(unbalanced)//tree unbalanced,determine rotation type
    {
        if(d==1)
        {    //left imbalance
           if(b->bf==1)     //rotation type LL
           {
                a->left=b->right;
                b->right=a;
                a->bf=0;
                b->bf=0;
           }
           else   //rotation type LR
           {
                c=b->right;
                b->right=c->left;
                a->left=c->right;
                c->left=b;
                c->right=a;
                switch(c->bf)
                {
                   case 1:a->bf=-1;  //LR(b)
                            b->bf=0;
                            break;
                   case -1:b->bf=1;  //LR(c)
                             a->bf=0;
                             break;
                   case 0:b->bf=0;  //LR(a)
                            a->bf=0;
                            break;
                }
                c->bf=0;
                b=c; //b is the new root
            } //end of LR
        }       //end of left imbalance
        else   //right imbalance
        {
           if(b->bf==-1)     //rotation type RR
           {
                a->right=b->left;
                b->left=a;
                a->bf=0;
                b->bf=0;
           }
           else   //rotation type LR
           {
                c=b->right;
                b->right=c->left;
                a->right=c->left;
                c->right=b;
                c->left=a;
                switch(c->bf)
                {
                    case 1:a->bf=-1;  //LR(b)
                             b->bf=0;
```

```
                                    break;
                                case -1:b->bf=1;  //LR(c)
                                        a->bf=0;
                                        break;
                                case 0:b->bf=0;  //LR(a)
                                        a->bf=0;
                                        break;
                                }
                                c->bf=0;
                                b=c; //b is the new root
                            } //end of LR
                        }
                    if(!f)
                            root=b;
                        else if(a==f->left)
                            f->left=b;
                        else if(a==f->right)
                            f->right=b;
                    }  //end of if unbalanced
                    return TRUE;
    }       //end of if(!found)
    return FALSE;
}    //end of AVL INSERTION
void AVL::removeitem(int x)
{
     search(x);
     if(loc==NULL)
     {
             cout<<"\nitem is not in tree";
             return;
     }
     if(loc->right!=NULL&&loc->left!=NULL)
     {
             remove1(loc,par,x);
        cout<<"\nitem is deleted";
     }
     else
     {
        remove2(loc,par,x);
        cout<<"\nitem is deleted";
     }
}
void AVL::remove1(AVLNODE *l,AVLNODE *p,int x)
{
     AVLNODE *ptr,*save,*suc,*psuc;
     ptr=l->right;
     save=l;
     while(ptr->left!=NULL)
     {
             save=ptr;
             ptr=ptr->left;
     }
     suc=ptr;
     psuc=save;
     remove2(suc,psuc,x);
     if(p!=NULL)
             if(l==p->left)
```

```
                p->left=suc;
        else
                p->right=suc;
    else
        root=l;
    suc->left=l->left;
    suc->right=l->right;
    return;
}
void AVL::remove2(AVLNODE *s,AVLNODE *p,int x)
{
    AVLNODE *child;
    if(s->left==NULL && s->right==NULL)
        child=NULL;
    else if(s->left!=NULL)
        child=s->left;
    else
        child=s->right;
    if(p!=NULL)
        if(s==p->left)
                p->left=child;
        else
                p->right=child;
    else
        root=child;
}
void AVL::search(int x)
{
    search1(root,x);
}
void AVL::search1(AVLNODE *temp,int x)
{
    AVLNODE *ptr,*save;
    int flag;
    if(temp==NULL)
    {
        cout<<"\nthe tree is empty";
        return;
    }
    if(temp->data==x)
    {
        cout<<"\nthe item is root and is found";
        par=NULL;
        loc=temp;
        par->left=NULL;
        par->right=NULL;
        return;
    }
    if(x<temp->data)
    {
        ptr=temp->left;
        save=temp;
    }
    else
    {
        ptr=temp->right;
        save=temp;
```

```
    }
    while(ptr!=NULL)
    {
        if(x==ptr->data)
        {
       flag=1;
          cout<<"\nitemfound";
          loc=ptr;
          par=save;
        }
        if(x<ptr->data)
          ptr=ptr->left;
        else
          ptr=ptr->right;
    }
    if(flag!=1)
    {
        cout<<"item is not there in tree";
        loc=NULL;
        par=NULL;
        cout<<loc;
        cout<<par;
    }
}
int main()
{
   AVL a;
   int x,y,c;
   char ch;
   do
   {
        cout<<"\n1.insert 2.delete 3.search 4.exit ";
        cout<<"\nEnter u r choice to perform on AVL tree";
        cin>>c;
        switch(c)
        {
          case 1:cout<<"\nEnter an element to insert into tree";
                 cin>>x;
                 a.insert(x);
                 break;
          case 2:cout<<"\nEnter an item to deletion";
                 cin>>y;
                 a.removeitem(y);
                 break;
          case 3:cout<<"\nEnter an element to search";
                 cin>>c;
                 a.search(c);
                 break;
          case 4:exit(0);
           break;
          default :cout<<"\nInvalid option try again";
        }
        cout<<"\nDo u want to continue:";
        cin>>ch;
   }while(ch=='y'||ch=='Y');
}
```

output:

```
vaag@vaag-HP-dx2480-MT-KL969AV: ~/rajitha
vaag@vaag-HP-dx2480-MT-KL969AV:~/rajitha$ g++ avl2.cpp
vaag@vaag-HP-dx2480-MT-KL969AV:~/rajitha$ ./a.out

1.insert 2.delete 3.search 4.exit
Enter u r choice to perform on AVL tree1

Enter an element to insert into tree11

Do u want to continue:y

1.insert 2.delete 3.search 4.exit
Enter u r choice to perform on AVL tree1

Enter an element to insert into tree12

Do u want to continue:y

1.insert 2.delete 3.search 4.exit
Enter u r choice to perform on AVL tree1

Enter an element to insert into tree13
```

```
vaag@vaag-HP-dx2480-MT-KL969AV: ~/rajitha

Do u want to continue:y

1.insert 2.delete 3.search 4.exit
Enter u r choice to perform on AVL tree1

Enter an element to insert into tree13

Do u want to continue:y

1.insert 2.delete 3.search 4.exit
Enter u r choice to perform on AVL tree3

Enter an element to search11

itemfound
Do u want to continue:y

1.insert 2.delete 3.search 4.exit
Enter u r choice to perform on AVL tree3

Enter an element to search14
item is not there in tree00
Do u want to continue:
```

```
vaag@vaag-HP-dx2480-MT-KL969AV: ~/rajitha
Do u want to continue:y

1.insert 2.delete 3.search 4.exit
Enter u r choice to perform on AVL tree3

Enter an element to search14
item is not there in tree00
Do u want to continue:y

1.insert 2.delete 3.search 4.exit
Enter u r choice to perform on AVL tree2

Enter an item to deletion11

itemfound
item is deleted
Do u want to continue:y

1.insert 2.delete 3.search 4.exit
Enter u r choice to perform on AVL tree3

Enter an element to search11
item is not there in tree00
Do u want to continue:
```

14. Write a C++ program using class templates to implement Kruskals algorithm to generate a minimum cost spanning tree.

```
#include<iostream>
#include<stdlib.h>
using namespace std;
int i,j,k,a,b,u,v,n,ne=1;
int mi,mincost=0,cost[9][9],parent[9];
int find(int);
int uni(int,int);
int main()
{
   cout<<"\n\tImplementation of Kruskal's algorithm\n";
   cout<<"\nEnter the no. of vertices:";
   cin>>n;
   cout<<"\nEnter the cost adjacency matrix:\n";
   for(i=1;i<=n;i++)
   {
         for(j=1;j<=n;j++)
         {
            cin>>cost[i][j];
            if(cost[i][j]==0)
                  cost[i][j]=999;
         }
   }
   cout<<"The edges of Minimum Cost Spanning Tree are\n";
   while(ne < n)
   {
         for(i=1,mi=999;i<=n;i++)
         {
            for(j=1;j<=n;j++)
            {
                  if(cost[i][j]<mi)
                  {
                     mi=cost[i][j];
                     a=u=i;
                     b=v=j;
                  }
            }
         }
         u=find(u);
         v=find(v);
         if(uni(u,v))
         {
            cout<<"\n"<<ne++<<" "<<"edge"<<" "<<"("<<a<<","<<b<<")"<<"="<<mi;
            mincost +=mi;
         }
         cost[a][b]=cost[b][a]=999;
   }
   cout<<"\n\tMinimum cost = "<<mincost;
}
```

```
int find(int i)
{
   while(parent[i])
         i=parent[i];
   return i;
}
int uni(int i,int j)
{
   if(i!=j)
   {
         parent[j]=i;
         return 1;
   }
   return 0;
}
```

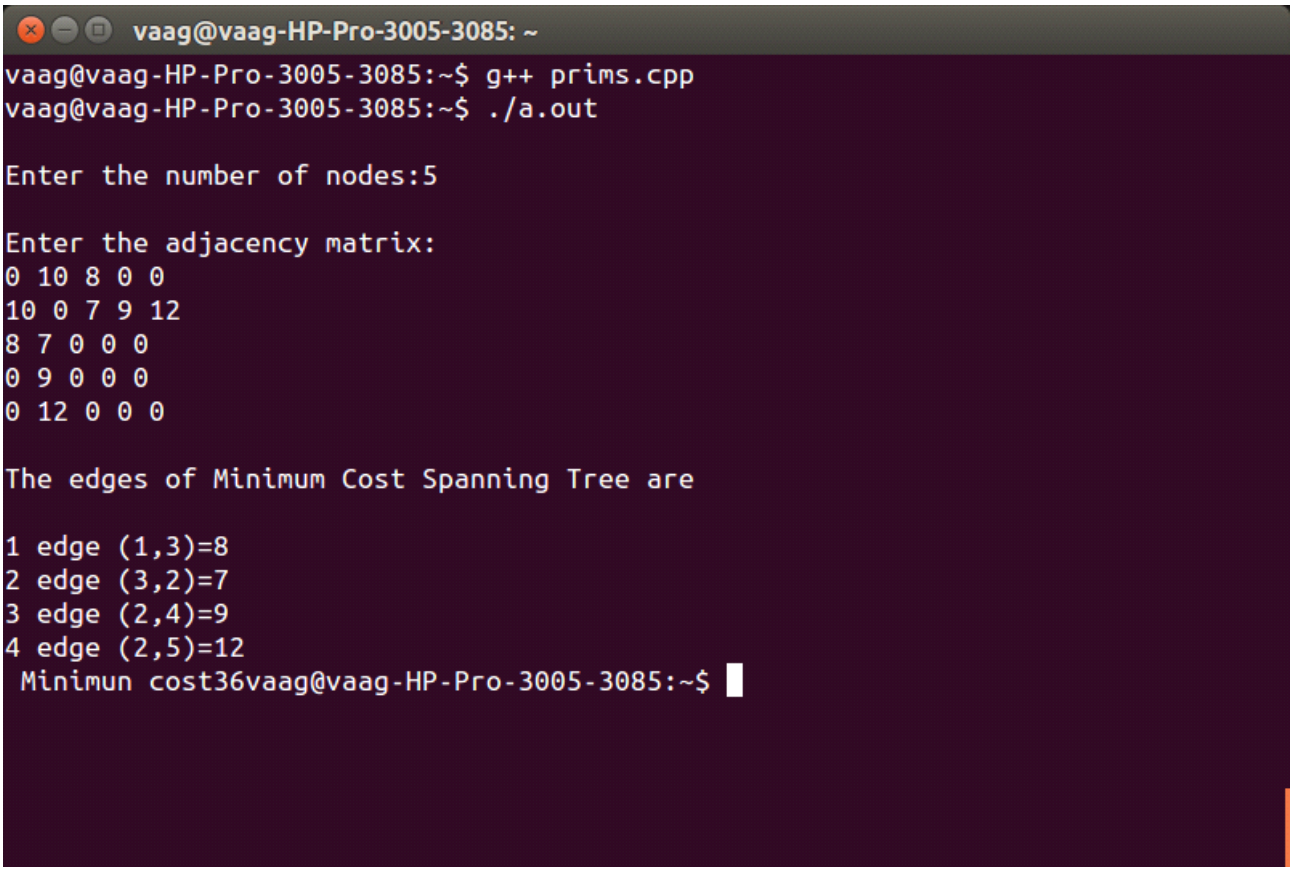
output:

```
vaag@vaag-HP-Pro-3005-3085: ~
vaag@vaag-HP-Pro-3005-3085:~$ g++ kruskal.cpp
vaag@vaag-HP-Pro-3005-3085:~$ ./a.out

        Implementation of Kruskal's algorithm

Enter the no. of vertices:5

Enter the cost adjacency matrix:
0 10 8 0 0
10 0 7 9 12
8 7 0 0 0
0 9 0 0 0
0 12 0 0 0
The edges of Minimum Cost Spanning Tree are

1 edge (2,3)=7
2 edge (1,3)=8
3 edge (2,4)=9
4 edge (2,5)=12
        Minimum cost = 36vaag@vaag-HP-Pro-3005-3085:~$
```

15. Write a C++ program using class templates to implement Prims algorithm to generate a minimum cost spanning tree.

```
#include<iostream>
#include<stdlib.h>
using namespace std;
int a,b,u,v,n,i,j,ne=1;
int visited[10]={0},mi,mincost=0,cost[10][10];
int main()
{
   cout<<"\nEnter the number of nodes:";
   cin>>n;
   cout<<"\nEnter the adjacency matrix:\n";
   for(i=1;i<=n;i++)
   for(j=1;j<=n;j++)
   {
          cin>>cost[i][j];
          if(cost[i][j]==0)
             cost[i][j]=999;
   }
   visited[1]=1;
   cout<<"\n";
   cout<<"The edges of Minimum Cost Spanning Tree are\n";
   while(ne < n)
   {
          for(i=1,mi=999;i<=n;i++)
          for(j=1;j<=n;j++)
          if(cost[i][j]< mi)
          if(visited[i]!=0)
          {
              mi=cost[i][j];
              a=u=i;
              b=v=j;
          }
          if(visited[u]==0 || visited[v]==0)
          {
              cout<<"\n"<<ne++<<" "<<"edge"<<" "<<"("<<a<<","<<b<<")"<<"="<<mi;
              mincost+=mi;
              visited[b]=1;
          }
          cost[a][b]=cost[b][a]=999;
   }
   cout<<"\n Minimun cost"<<mincost;
}
```

output:


vaag@vaag-HP-Pro-3005-3085: ~
vaag@vaag-HP-Pro-3005-3085:~$ g++ prims.cpp
vaag@vaag-HP-Pro-3005-3085:~$ ./a.out
Enter the number of nodes:5
Enter the adjacency matrix:
0 10 8 0 0
10 0 7 9 12
8 7 0 0 0
0 9 0 0 0
0 12 0 0 0
The edges of Minimum Cost Spanning Tree are
1 edge (1,3)=8
2 edge (3,2)=7
3 edge (2,4)=9
4 edge (2,5)=12
Minimun cost36vaag@vaag-HP-Pro-3005-3085:~$

16. Write a c++ program to implement Knuth-Morris-Pratt pattern matching algorithm.

```
#include<iostream>
#include<string.h>
#include<stdlib.h>
int n,m,fail[256];
char t[512],p[256]="hai";
using namespace std;
void failurefunction()
{
    int i,j;
    fail[0]=0;
    m=strlen(p);
    j=0;
    i=1;
    while(i<m)
    {
       if(p[j]==p[i])
       {
          fail[i]==j+1;
          i++;
          j++;
       }
       else
       {
          if(j>0)
             j=fail[j-1];
          else
          {
             fail[i]=0 ;
             i++;
          }
       }
    }
}
int main()
{
    char ch;
    int i,j,flag=0;
    cout<<"\nenter text(at end press dot(.)):\n";
    i=0;
    while(ch!='.')
    {
       cin>>ch;
       t[i]=ch;
       i++;
    }
    t[i]='\0';
    cout<<"\ntext is:\n"<<t;
    cout<<"\npattern is:\n"<<p;
    n=strlen(t);
```

```
    m=strlen(p);
    failurefunction();
    i=j=0;
    while(i<n)
    {
       if(p[j]==t[i])
       {
          if(j==(m-1))
          {
             cout<<"\npattern found at:"<<(i-m+1);
             flag=1;
          }
          i++;
          j++;
       }
       else
       {
          if(j>0)
             j=fail[j-1];
          else i++;
       }
    }
    if(flag==0)
    {
       cout<<"\n pattern not found";
    }
}
```

output: