NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

PROGRAM: SOFTWARE ENGINEERING

DATA STRUCTURES LAB

LAB TASK 10

SUBMITTED BY:

Name: Ahmed Ali

Roll No: 22P-9318

SECTION: BS (SE) 4-C

INSTRUCTOR NAME: Sir Saood Sarwar

SUBMISSION DEADLINE: 31/03/2024

A DEPARTMENT OF COMPUTER SCIENCE

QUESTION NO.1 CODE:

```
#include<iostream>

using namespace std;


class ln

{

        public:

            string address;

            int price;

            ln *next;


            ln(int price, string address)

                {

            this->price=price;

            this->address=address;

            this->next=nullptr;

            }

};


class node

{

        public:

            int price;

            string address;

            node *left;

            node *right;


            node(int price, string address)
```

```
        {
    this->price=price;
    this->address=address;
    this->left=nullptr;
    this->right=nullptr;
  }
};

class bst
{
    private:
  node *root;

    public:
        bst()
        {
            root=nullptr;
        }

  node *insert(node *n, int price, string address)
        {
    if(n==nullptr)
            {
      return new node(price, address);
    }
    if(price<n->price)
            {
      n->left=insert(n->left, price, address);
    }
```

```
                else
                {
        n->right=insert(n->right, price, address);
    }
    return n;
}


        void insert(int price, string address)
        {
    root=insert(root, price, address);
}


node *search(node *n, int price)
        {
    if(n==nullptr)
                {
        return nullptr;
    }
    if(n->price==price)
                {
        return n;
    }
    if(price<n->price)
                {
        return search(n->left, price);
    }
                else
                {
        return search(n->right, price);
```

```
    }
}

node *search(int price)
        {
    return search(root, price);
}

void in_order(node *n)
        {
    if(n==nullptr)
                {
        return;
    }
    in_order(n->left);
    cout<<n->price<<"\t"<<n->address<<endl;
    in_order(n->right);
}

        void in_order()
        {
    in_order(root);
    cout<<endl;
}

void pre_order(node *n)
        {
    if(n==nullptr)
                {
```

```
        return;
    }
                    cout<<n->price<<"\t"<<n->address<<endl;
    pre_order(n->left);
    pre_order(n->right);
}

        void pre_order()
        {
    pre_order(root);
    cout<<endl;
}

void post_order(node *n)
        {
    if(n==nullptr)
                    {
        return;
    }
    post_order(n->left);
    post_order(n->right);
    cout<<n->price <<"\t"<<n->address<<endl;
}

        void post_order()
        {
    post_order(root);
    cout<<endl;
}
```

```
node *find_min(node *n)
        {
   while(n->left!=nullptr)
                    {
      n=n->left;
   }
   return n;
}

int find_min()
        {
   node *min=find_min(root);
   if(min!=nullptr)
                    {
      return min->price;
   }
                    else
                    {
      return -1;
   }
}

node *find_max(node *n)
        {
   while(n->right!=nullptr)
                    {
      n=n->right;
   }
```

```
    return n;
}

int find_max()
        {
    node *max=find_max(root);
    if(max!=nullptr)
    {
        return max->price;
    }
                    else
                    {
        return -1;
    }
}

int maximum(int a, int b)
        {
    if(a>b)
                    {
        return a;
    }
                    else
                    {
        return b;
    }
}

int height(node *n)
```

```
        {
    if(n==nullptr)
                    {
        return -1;
    }
    int left_height=height(n->left);
    int right_height=height(n->right);
    if(left_height>right_height)
                    {
        return left_height+1;
    }
                    else
                    {
        return right_height+1;
    }
}

int height()
        {
    return height(root);
}

void above_price(node *n, int price, ln *&head)
        {
    if (n==nullptr)
                    {
        return;
    }
    if(n->price > price)
```

```
                {
        ln* new_node=new ln(n->price, n->address);
        new_node->next=head;
        head=new_node;
    }
    above_price(n->left, price, head);
    above_price(n->right, price, head);
}

ln *above_price(int price)
        {
    ln *head=nullptr;
    above_price(root, price, head);
    return head;
}

void below_price(node *n, int price, ln *&head)
        {
    if(n==nullptr)
                {
        return;
    }
    if(n->price<price)
                {
        ln *new_node=new ln(n->price, n->address);
        new_node->next=head;
        head=new_node;
    }
    below_price(n->left, price, head);
```

```
    below_price(n->right, price, head);
}

ln *below_price(int price)
        {
    ln *head=nullptr;
    below_price(root, price, head);
    return head;
}

void in_range(node *n, int min_price, int max_price, ln *&head)
        {
    if(n==nullptr)
                {
        return;
    }
    if(n->price>=min_price && n->price<=max_price)
                {
        ln *new_node=new ln(n->price, n->address);
        new_node->next=head;
        head=new_node;
    }
    in_range(n->left, min_price, max_price, head);
    in_range(n->right, min_price, max_price, head);
}

        ln *in_range(int min_price, int max_price)
        {
    ln *head=nullptr;
```

```
        in_range(root, min_price, max_price, head);
        return head;
    }

    void by_address(node *n, string address, ln *&head)
        {
        if(n==nullptr)
                {
            return;
        }
        if(n->address==address)
                {
            ln *new_node=new ln(n->price, n->address);
            new_node->next=head;
            head=new_node;
        }
        by_address(n->left, address, head);
        by_address(n->right, address, head);
    }

    ln *by_address(string address)
        {
        ln *head=nullptr;
        by_address(root, address, head);
        return head;
    }
};

int main()
```

```
{
    bst T1;

    T1.insert(22000, "A");
    T1.insert(52000, "B");
    T1.insert(23000, "C");
    T1.insert(41000, "D");
    T1.insert(37000, "E");
    cout<<"inorder:"<<endl;
    T1.in_order();
    cout<<"preorder:"<<endl;
    T1.pre_order();
    cout<<"postorder:"<<endl;
    T1.post_order();

    int search_price=41000;
    if(T1.search(search_price))
    {
        cout<<"property with price "<<search_price<<"founded"<<endl;
    }
            else
            {
        cout<<"property with price "<<search_price<<"not in the record"<<endl;
    }
    cout<<"min price property: "<<T1.find_min()<<endl;
    cout<<"max price property: "<<T1.find_max()<<endl;
    cout<<"height: "<<T1.height()<<endl;

    int above_price=230000;
```

```
ln *above_list=T1.above_price(above_price);
cout<<"properties above given price "<<above_price<<":"<<endl;
while(above_list!=nullptr)
        {
   cout<<above_list->price<<"\t"<<above_list->address<<endl;
   above_list=above_list->next;
}

int below_price=52000;
ln *below_list=T1.below_price(below_price);
cout<<"properties below price "<<below_price<<":"<<endl;
while(below_list!=nullptr)
        {
   cout<<below_list->price<<"\t"<<below_list->address<<endl;
   below_list=below_list->next;
}

int min_price=1000;
int max_price=50000;
ln *range_list=T1.in_range(min_price, max_price);
cout<<"properties in price from"<<min_price<<"-"<<max_price<<":"<<endl;
while(range_list!=nullptr)
        {
   cout<<range_list->price<<"\t"<<range_list->address<<endl;
   range_list=range_list->next;
}

string address_search="XYZ";
ln *address_list=T1.by_address(address_search);
```

```
    cout<<"properties at address "<<address_search<<":"<<endl;
    while(address_list!=nullptr)
        {
      cout<<address_list->price<<"\t"<<address_list->address<<endl;
      address_list=address_list->next;
    }
    return 0;
}
```

Output-01:

```
D:\SUMMER' 24\Data Structures LAB\LAB TASK 10 Sibing etc\lab task\lab task 10.ex
inorder:
22000   A
23000   C
37000   E
41000   D
52000   B

preorder:
22000   A
52000   B
23000   C
41000   D
37000   E

postorder:
37000   E
41000   D
23000   C
52000   B
22000   A

property with price 41000founded
min price property: 22000
max price property: 52000
height: 4
properties above given price 230000:
properties below price 52000:
37000   E
41000   D
23000   C
22000   A
properties in price from1000-50000:
37000   E
41000   D
23000   C
22000   A
properties at address XYZ:

--------------------------------
Process exited after 0.6657 seconds with return value 0
Press any key to continue . . .
```