```
1: //                                          Assignment no-
   6
2: // Name- Prerana Rajesh Gajare      Class-SEIT
   RollNo-SI41
3: /*PROBLEM STATEMENT:-
4:              Implement In-order Threaded Binary Tree
   and traverse it in In-order and Pre-order.
5:  */
6: //Source Code:-
7: #include <iostream>
8: using namespace std;
9:
10: //Class tbt
11: class tbt
12:  {
13:     int data;//Integer data
14:     tbt *left,*right;//Left and Right address pointer
15:     bool rthread,lthread;//Left and right thread
16:     public:
17:         //Function declaration
18:         tbt* insert(tbt *,tbt *);
19:         tbt* create(int);
20:         tbt* leftmostnode(tbt *,tbt *);
21:         void inorder(tbt *,tbt *);
22:         tbt* preorder(tbt *,tbt *);
23:
24:  };
25:
26:  tbt* tbt :: create(int key)
27:  {
28:     tbt* newnode=new tbt;//Create newnode using new
   keyword
29:     newnode->data=key;//Store key value in newnode of
   data
30:     newnode->left=newnode;//Store newnode vlaue in
   newnode's left and right
31:     newnode->right=newnode;
```

```
32:     newnode->lthread=true;//Initialize leftthread as
    true
33:     newnode->rthread=true;//Initialize rightthread as
    true
34:     return newnode;
35:  }
36:
37:  // TO PERFORM INSERT OPERATION IN TBT
38:  tbt* tbt :: insert(tbt *root,tbt *head)
39:  {
40:     int key;
41:     tbt *curr;
42:     //Accepting the element to be inserted
43:     cout<<"Enter the element to insert:";
44:     cin>>key;
45:     //Creating a node for element inserted using
    create function
46:     tbt *newnode=create(key);
47:
48:     //If tree is empty
49:     if(root==NULL)
50:     {
51:         head->left=newnode;//Set newnode to left of
    head
52:         root=newnode;//Store newnode value in root node
53:         root->left=root->right=head;
54:         head->lthread=false;
55:         return root;
56:     }
57:     else
58:     {
59:         curr=root;//Set root as curr
60:         while(1)
61:         {
62:             // To check if the data is already present
    or not.
63:             if(curr->data==key)
```

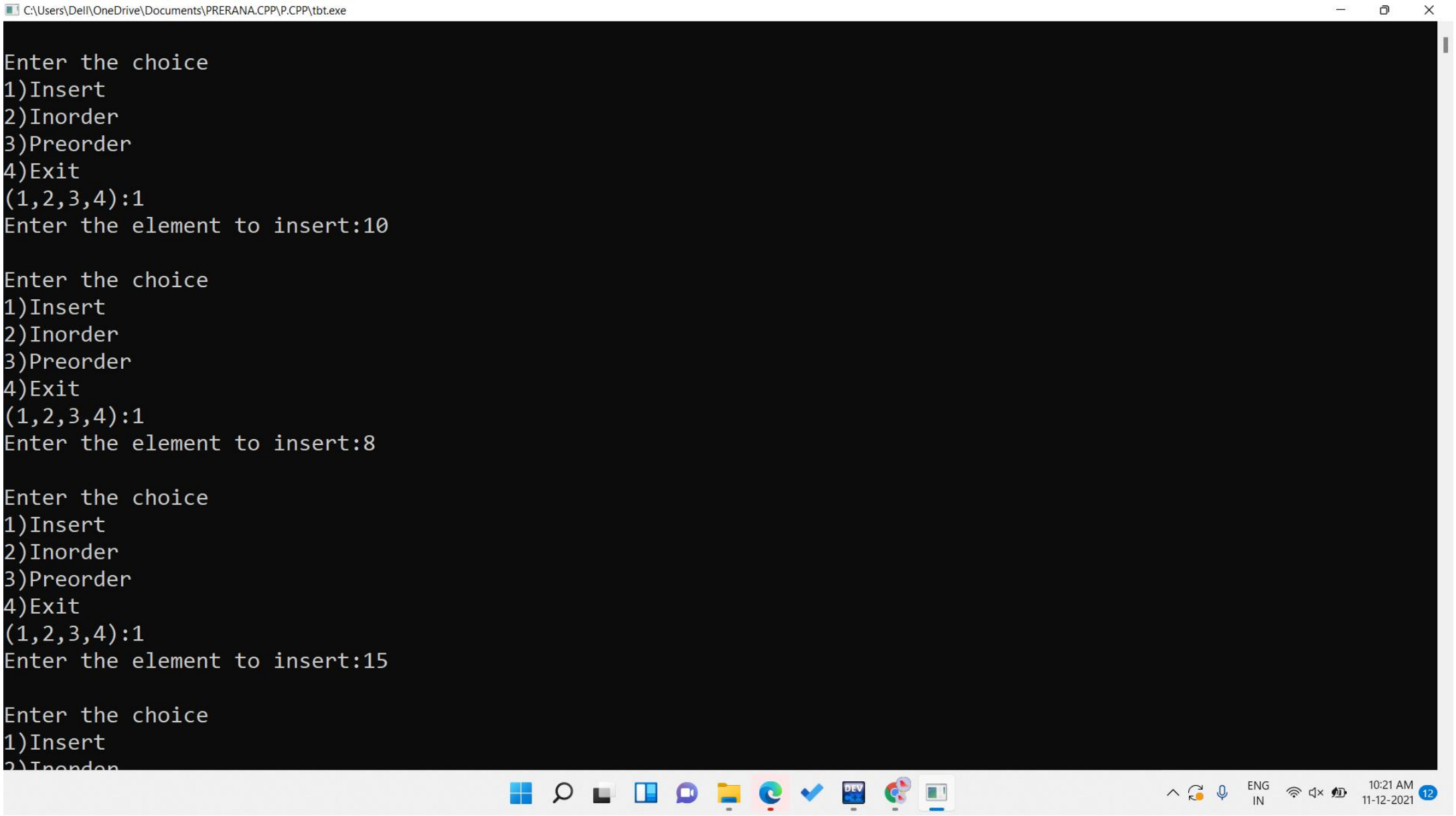
```
64:             {
65:                 cout<<" Element is present ";
66:                 return root;;
67:                 break;
68:             }
69:              //If data(key) is less than curr
70:             if(key<curr->data)
71:             {
72:                 // If curr of leftthread is true
73:                 if(curr->lthread)
74:                 {
75:                     newnode->left=curr->left;
76:                     curr->left=newnode;//Store the
    value of newnode in curr of left
77:                     newnode->right=curr;//Set curr in
    newnode's right
78:                     curr->lthread=false;//Set curr
    left thread as false
79:                     break;
80:                 }
81:                 else
82:                 {
83:                     //Move towards left
84:                     curr=curr->left;
85:                 }
86:             }
87:             else
88:             {
89:                 // If curr of rightthread is true
90:                 if(curr->rthread)
91:                 {
92:                     newnode->left=curr;//Set curr in
    newnode's left
93:                     newnode->right=curr->right;
94:                     curr->right=newnode; //Store the
    value of newnode in curr of right
95:                     curr->rthread=false;//Set curr
    right thread as false
```

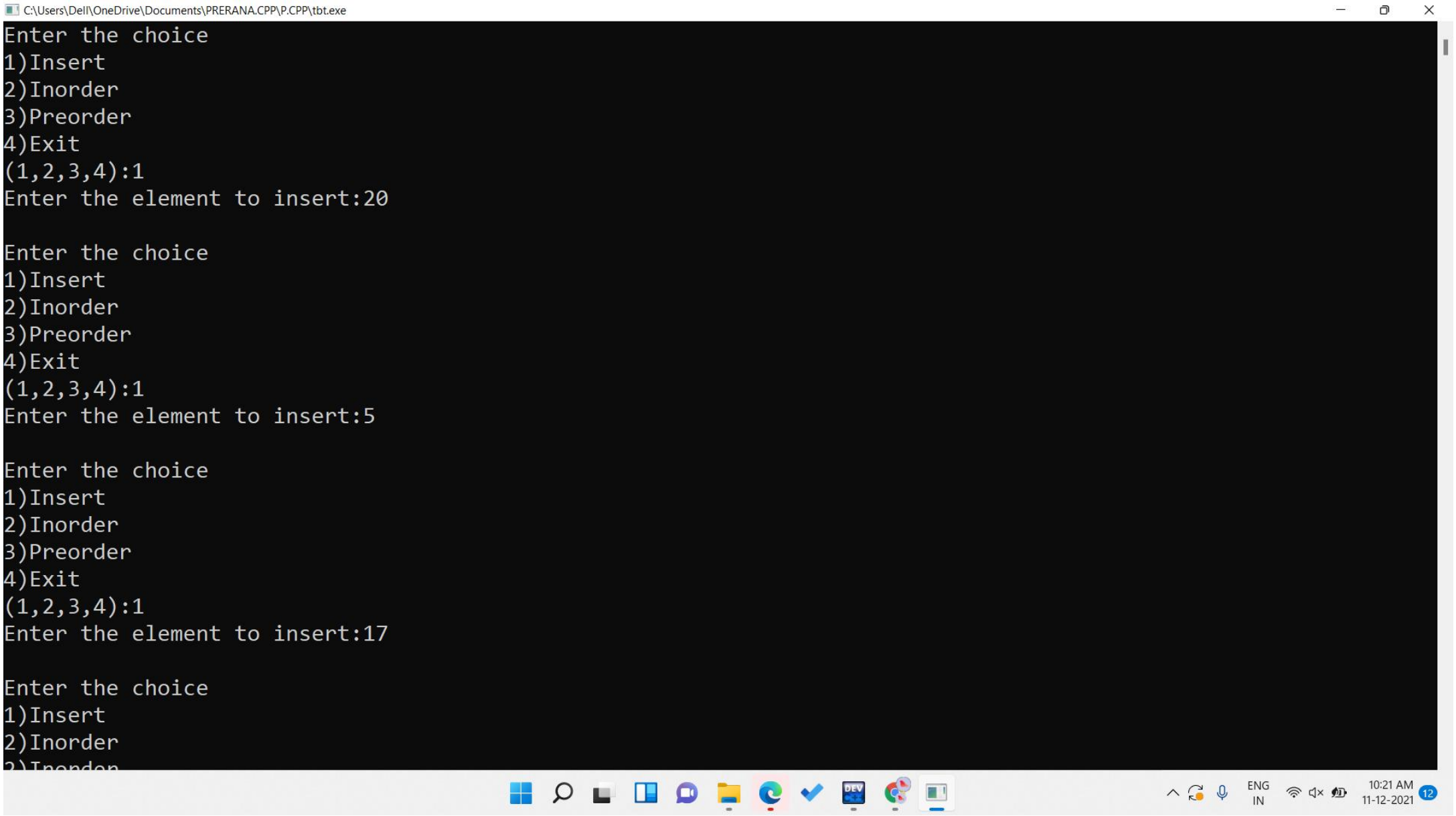
```
 96:                     break;
 97:                 }
 98:                 else
 99:                 {
100:                     //Move towards right
101:                     curr=curr->right;
102:                 }
103:             }
104:         }
105:     }
106:     return root;
107:  }
108:
109:  // To reach to the leftmostnode of the tree
110: tbt *tbt :: leftmostnode(tbt *temp,tbt *head)
111:  {
112:     // If tree is empty return NULL
113:     if(temp==NULL)
114:     {
115:         return NULL;
116:     }
117:     else
118:     {
119:         while(temp->left!=head)
120:         {
121:
122:             if(temp->lthread)
123:             {
124:                 break;
125:             }
126:             temp=temp->left;
127:
128:         }
129:         return temp;
130:     }
131:  }
132:
```

```
133:  //To perform inorder traversal(LVR) of tbt
134: void tbt :: inorder(tbt *root,tbt *head)
135: {
136:     tbt * curr;
137:     curr=leftmostnode(root,head);//Store temp value in
     curr
138:     while(curr!=head)
139:     {
140:         cout<<"\t"<<curr->data;
141:         //If right thread is true
142:         if(curr->rthread)
143:         {
144:         //Move towards right
145:         curr=curr->right;
146:         }
147:         else
148:         {
149:
150:         curr=leftmostnode(curr->right,head);
151:         }
152:     }
153: }
154:
155: //To perform preorder traversal(VLR) of tbt
156: tbt* tbt :: preorder(tbt *root,tbt *head)
157: {
158:     tbt *curr=root;//Store root value in curr
159:     while(1)
160:     {
161:         cout<<"\t"<<curr->data;
162:
163:         if(curr->lthread==false)//If Actual child
     present move to left
164:             curr= curr->left;
165:         else if(curr->rthread==false)//If Actual child
     present move to right
166:             curr=curr->right;
```

```
167:         else
168:         {
169:             while(curr->right!=head && curr-
    >rthread==true)
170:             {
171:                 curr=curr->right;
172:             }
173:             if(curr->right!=head)
174:                 curr=curr->right;
175:             else
176:                 break;
177:         }
178:
179:     }
180:  }
181:
182:  int main()
183:  {
184:     tbt *root= NULL;//Initalize root as NULL
185:     int l;
186:     tbt t;
187:     tbt *head=t.create(999);
188:  do
189:  {
190:     cout<<"\nEnter the
    choice\n1)Insert\n2)Inorder\n3)Preorder\n4)Exit\n(1,2,
    3,4):";
191:     cin>>l;
192:   elseitch
193:     {
194:         case 1:
195:             root=t.insert(root,head);//Calling insert
    function and storing its value in root
196:             break;
197:         case 2:
198:             t.inorder(root,head);//Calling inorder
    function
```

```
199:                 break;
200:             case 3:
201:                 t.preorder(root,head);//Calling preorder
    function
202:                 break;
203:             case 4:
204:                 cout<<"The end";
205:                 break;
206:
207:         }
208:         }while(l!=4);
209:
210:  }
```


C:\Users\Dell\OneDrive\Documents\PRERANA.CPP\P.CPP\tbt.exe
Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
(1,2,3,4):1
Enter the element to insert:10
Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
(1,2,3,4):1
Enter the element to insert:8
Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
(1,2,3,4):1
Enter the element to insert:15
Enter the choice
1)Insert
ENG
IN
10:21 AM
11-12-2021


C:\Users\Dell\OneDrive\Documents\PRERANA.CPP\P.CPP\tbt.exe
Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
(1,2,3,4):1
Enter the element to insert:20
Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
(1,2,3,4):1
Enter the element to insert:5
Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
(1,2,3,4):1
Enter the element to insert:17
Enter the choice
1)Insert
2)Inorder
ENG
IN
10:21 AM
11-12-2021

C:\Users\Dell\OneDrive\Documents\PRERANA.CPP\P.CPP\tbt.exe

```
Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
(1,2,3,4):1
Enter the element to insert:7

Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
(1,2,3,4):2
       5       7       8       10      15      17      20
Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
(1,2,3,4):3
       10      8       5       7       15      20      17
Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
2)Inorder
```

C:\Users\Dell\OneDrive\Documents\PRERANA.CPP\P.CPP\tbt.exe

```
3)Preorder
4)Exit
(1,2,3,4):1
Enter the element to insert:7

Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
(1,2,3,4):2
       5       7       8       10      15      17      20
Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
(1,2,3,4):3
       10      8       5       7       15      20      17
Enter the choice
1)Insert
2)Inorder
3)Preorder
4)Exit
(1,2,3,4):4
The end
--------------------------------
2)Inorder
```