```
AVL_Trees.c ₽
   \leftarrow
                                                           \overline{\geq}
                                                                  :
            Saved
    #include <stdio.h>
2
    typedef struct node
       int data;
       struct node *left, *right;
       int ht;
9
10
   node;
11
   node* insert(node *, int);
   node* Delete(node *,
14
    void preorder(node*);
   void inorder(node*);
    int height(node*);
17
   node* rotateright(node*);
18 node* rotateleft(node*);
   node* RR(node*);
20 node* LL(node*)
21
   node* LR(node*);
22
   node* RL(node*);
23
    int BF(node*);
24
25
   int main()
26
27
       node *root = NULL;
28
       int x, n, i, op;
29
                   "\n1)Create:");
"\n2)Insert:");
"\n3)Delete:");
"\n4)Print:");
         printf(
30
         printf(
         printf(
33
         printf(
         printf("\n5)
scanf("%d"
34
         printf(
35
                                       Choice:");
                         %op);
36
         switch (op)
37
38
39
              printf("\nEnter
scanf("%d", &n);
printf("\nEnter
40
41
42
               root = NULL;
43
44
                             i < n; i++)
45
               {
                 scanf("%d", &x);
root = insert(root, x);
46
47
48
49
50
51
              printf("\nEnter
scanf("%d", &x);
52
               root =
                       insert(root, x);
54
56
              printf("\nEnter
scanf("%d", &x);
57
58
              root = Delete(root, x);
break;
59
60
61
62
63
              preorder(root);
64
65
              inorder(root);
66
67
               break;
68
       } while (op != 5);
69
70
71
```

72 73 74 75 76 77 78 79 88	nc {	ode* insert(node *T, int x)
75 76		if (T == NULL)
77 78		<pre>T = (node*) malloc(sizeof(node)); T->data = x; T->left = NULL; T->right = NULL;</pre>
79 80 81		
		<pre>} else if (x > T->data) // insert in right subtree { T->right = insert(T->right, x); }</pre>
83 84 85		<pre>{ T->right = insert(T->right, x);</pre>
86 87 88 89		<pre>T->right = insert(T->right, x); if (BF(T) == -2) if (x > T->right->data) T = RR(T);</pre>
89 90		
98 91 92 93		} else if (x < T->data)
93 94 95		
94 95 96 97 98		<pre>T->left = insert(T->left, x); if (BF(T) == 2) if (x < T->left->data) T = LL(T);</pre>
98 99		T = LL(T); else
100) = LR(1);
103 104		T->ht = height(T); return (T);
105 106	}	T = LL(T); clas T = LR(T); } T-btt = height(T); recurn (T); dde+ Delete(node *T, 3nt x) node *p; if (T == NULL) { cturn NUL; } ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ; ;
108 109	{	node * p;
118 111		if (T == NULL) {
112 113 114		return NULL; }
115 116		/
115 116 117 118 119 120 121 122		<pre>T->right = Delete(T->right, x); if (BF(T) == 2) if (BF(T->left) >= 8) T = LL(T);</pre>
128		
122 123		
123 124 125 126 127 128 129 130 131 132 133 134 135		else if (x < T->data) {
127 128		<pre>T->left = Delete(T->left, x); if (BF(T) == -2) //Rebalance during windup if (BF(T->right) <= 0) T = RR(T);</pre>
129 130		if (BF(T->right) <= 0) T = RR(T);
131 132		T = PI (T).
134 135		
136 137 138		//data to be deleted is found if (T->right != NULL)
139		//delete its inorder successor p = T-oright; p = T-oright; p = p-left; p = p-left; r-deta p = p-left; r (GF(T) = 0) //Bebalance during windup if (GF(T) = 0) //Bebalance during windup if = LL(T); else
149 141 142 143 144 145 146		<pre>while (p->left != NULL) p = p->left;</pre>
143 144		T->data = p->data; T->right = Delete(T->right, p->data); if (BF(T) == 2) //Rebalance during windup
146 147		<pre>if (BF(T) == 2) //Rebalance during windup if (BF(T->left) >= 0) T = LL(T):</pre>
147 148 149 150 151		T = LR(T);\
150 151		} else return (T->left); }
152 153 154		
155 156 157 158 159		T->ht = height(T); return (T);
157 158 159		at height(node *T)
161 162 163 164		int lh, rh; if (T == NULL) return (0); if (T-zleft == NULL) lh = 0; elso
164 165 166 167		if (T->left == NULL) lh = 0;
167 168		elme
169 170		<pre>lh = ! + T-left-sht; if (T-sright == NAL) in = ! if (1-sright == NAL) if (1 h > rh) return (1h); return (1h); dee rotateright(node *x) node *y; y = x-sift; y = x-sift;</pre>
171 172 173		rh = 1 + T->right->ht; if (lh > rh) return (lh):
174 175		return (rh);
176 177	no	ode* rotateright(node *x)
179 189		<pre>node * y; y = x>left; y = x>left = y>right; y>right = x; x=hit = height(x); y=hit = height(y); return (y);</pre>
181 182 183 184		x->left = y->right; y->right = x;
183 184 185		<pre>x->ht = height(x); y->ht = height(y);</pre>
186 187		
188 189 198		ode* rotateleft(node *x)
198 191 192		<pre>node * y; y = x > x > right; y = x > right = y > left; y = x > left = x; x = left = x x y = y = left; y = left = height(y); return (y);</pre>
193 194		y->left = x; x->ht = height(x);
195 196		y->ht = height(y); return (y);
197 198 199) nr	ode+ PR(node +T)
288 281	{	T = rotateleft(T); return (T);
202 203		
205 206	nc {	ode* LL(node *T)
207 208		<pre>T = rotateright(T); return (T);</pre>
209 210 211) rs	ode* LR(node *T)
212 213		T->left = rotateleft(T->left);
214 215		<pre>T->left = rotateleft(T->left); T = rotateright(T); return (T);</pre>
216 217 218) no	ode* RL(node *T)
219 228		T->right = rotateright(T->right);
221 222		<pre>T->right = rotateright(T->right); T = rotateleft(T); return (T);</pre>
224 225		it BF(node *T)
226 227		int 1h, rh;
228 229 238		return (0);
231 232		<pre>if (T->left == NULL) lh = 0; else lh = 1 + T->left->ht;</pre>
233 234		else lh = 1 + T->left->ht;
235 236 237		if (T->right == NULL) rh = 0;
238 239		else rh = 1 + T->right->ht;
248 241		
242 243 244		oid preorder(node *T)
245 246		if (T != NULL)
247 248 240		<pre>{ printf("Md(Bf=Md)", T->data, BF(T)); preorder(T->left); preorder(T->right);</pre>
250 251		preorder(T->right); }
252 253		y-oht = height(y); return (y); de+ RE(node +T) T = rotateleft(T); return (T); de+ LL(node +T) T = rotateleft(T); return (T); de+ LL(node +T) T = rotateleft(T); return (T); de+ RE(node +T) T = rotateleft(T->left); return (T); de+ RL(node +T) T = rotateleft(T); return (T); it BF(node +T) in: h, rh; if (T == MALL) th = 0; th = i + T->left->ht; if (T->right == MALL) th = 0; th = i + T->right->ht; return (T) + return (T) + return (T) + return (T) if (T->right == MALL) th = 0; th = i + T->right->ht; return (T) + right->ht; return (T) + rig
254 255 256		if (T != NULL)
257 258		<pre>inorder(T->left); printf("\d(\Bf=\d)", T->data, BF(T));</pre>
259 260 261		<pre>printf("\d(Bf=\d)", T->data, BF(T)); inorder(T->right); }</pre>
261 262		

```
×
         Terminal
2)Insert:
3)Delete:
Enter Your Choice:1
Enter tree data:20 30 40 50 60
2) Insert:
3)Delete:
20(Bf=0)30(Bf=-1)40(Bf=0)50(Bf=0)60(Bf=0)
1)Create:
2)Insert:
3)Delete:
2)Insert:
3)Delete:
2)Insert:
3)Delete:
4)Print:
Enter Your Choice:4
15(Bf=0)20(Bf=1)30(Bf=-1)40(Bf=-1)45(Bf=0)50(Bf=1)60(Bf=0)
2)Insert:
3)Delete:
4)Print:
Enter a data:30
2)Insert:
Preorder sequence:
40(Bf=0)20(Bf=1)15(Bf=0)50(Bf=0)45(Bf=0)60(Bf=0)
```