

```

1: /*-----Assignment No - 4-----*/
2: #include<iostream>
3: #include<stdio.h>
4: #include<stdlib.h>
5: using namespace std;
6: typedef struct node
7: {   int a;
8:     struct node *left,*right;
9: }node;
10: struct node *root=NULL;
11: class BST
12: {
13:
14:     int count;
15:     struct node *temp=NULL,*t1=NULL,*s=NULL, *t=NULL;
16: public:
17:     BST()
18:     {
19:         count=0;
20:     }
21:     struct node *create();
22:     void insert();
23:     int height(struct node*,int c);
24:     int findmin(struct node*);
25:     void swap(struct node*);
26:     void search(struct node * root, int m);
27:     void display(struct node*);
28:
29: };
30:
31: int main()
32: {
33:     BST b;
34:     int x, m, c=0,cnt,min,fl;
35:     do
36:     {
37:         cout<<"\n Enter your choice:";
38:         cout<<"\n 1.insert:";
39:         cout<<"\n 2.No of nodes in longest path:";
40:         cout<<"\n 3.Minimum value:";
41:         cout<<"\n 4.Swap:";
42:         cout<<"\n 5.Search:";
43:         cout<<"\n 6.display:";
44:         cout<<"\n 7.exit:";
45:         cin>>x;
46:         switch(x)
47:         {
48:             case 1:
49:                 b.insert();
50:                 break;
51:             case 2:
52:                 cnt=b.height(root, c);
53:                 cout<<"\n No of nodes in longest path="<<cnt;
54:                 break;
55:             case 3:
56:                 min=b.findmin(root);
57:                 cout<<"\n Minimum value = "<<min;
58:                 break;
59:             case 4:
60:                 b.swap(root);
61:                 break;

```

```

62:         case 5:
63:             cout<<"\n enter data to be searched:";
64:             cin>>m;
65:             b.search(root, m);
66:             break;
67:         case 6 :
68:             b.display(root);
69:             break;
70:         case 7:
71:             exit(0);
72:     }
73: }while(x!=7);
74: return 0;
75: }
76:
77: struct node *BST::create()
78: {
79:     node *p=new(struct node);
80:     p->left=NULL;
81:     p->right=NULL;
82:     cout<<"\n enter the data";
83:     cin>>p->a;
84:     return p;
85: }
86:
87: void BST::insert()
88: {
89:     temp=create();
90:     if(root==NULL)
91:     {
92:         root=temp;
93:     }
94:     else
95:     {
96:         t1=root;
97:         while(t1!=NULL)
98:         {
99:             s=t1;
100:             if((temp->a)>(t1->a))
101:             {
102:                 t1=t1->right;
103:             }
104:             else
105:             {
106:                 t1=t1->left;
107:             }
108:         }
109:         if((temp->a)>(s->a))
110:         {
111:             s->right=temp;
112:         }
113:         else
114:         {
115:             s->left=temp;
116:         }
117:     }
118: }
119:
120: int BST::height(struct node *q,int c)
121: {
122:     if(root==NULL)

```

```

123:     {
124:         cout<<"\n tree not exist";
125:     }
126:     else
127:     {
128:         c++;
129:         if(q->left!=NULL)
130:         {
131:             height(q->left,c);
132:         }
133:         if(q->right!=NULL)
134:         {
135:             height(q->right,c);
136:         }
137:         if(count<c)
138:         {
139:             count=c;
140:         }
141:     }
142:     return count;
143: }
144:
145: int BST::findmin(node *T)
146: {
147:     while(T->left!=NULL)
148:     {
149:         T=T->left;
150:     }
151:     return T->a;
152: }
153:
154: void BST::swap(struct node *q)
155: {
156:     if(q==NULL)
157:     {
158:         cout<<"\n tree not exist";
159:     }
160:     else
161:     {
162:         if(q->left!=NULL)
163:             swap(q->left);
164:         if(q->right!=NULL)
165:             swap(q->right);
166:         t=q->left;
167:         q->left=q->right;
168:         q->right=t;
169:     }
170: }
171:
172: void BST::search(struct node * root, int m)
173: {
174:     int f;
175:     if(root!=NULL)
176:     {
177:         if(root->a==m)
178:             f=1;
179:         if(m>root->a)
180:             search(root->right,m);
181:         else
182:             search(root->left,m);
183:     }

```

```

184:     if(f==1)
185:         cout<<"\n FOUND!!!";
186: }
187:
188: void BST::display(struct node *q)
189: {
190:     if(q==NULL)
191:     {
192:         cout<<"\n tree not exist";
193:     }
194:     else
195:     {
196:         cout<<"\n*"<<q->a;
197:         if(q->left!=NULL)
198:         {
199:             display(q->left);
200:         }
201:         if(q->right!=NULL)
202:         {
203:             display(q->right);
204:         }
205:     }
206: }
207: /* OUTPUT
208:
209: Enter your choice:
210: 1.insert:
211: 2.No of nodes in Longest path:
212: 3.Minimum value:
213: 4.Swap:
214: 5.Search:
215: 6.display:
216: 7.exit:1
217:
218: enter the data56
219:
220: Enter your choice:
221: 1.insert:
222: 2.No of nodes in Longest path:
223: 3.Minimum value:
224: 4.Swap:
225: 5.Search:
226: 6.display:
227: 7.exit:1
228:
229: enter the data567
230:
231: Enter your choice:
232: 1.insert:
233: 2.No of nodes in Longest path:
234: 3.Minimum value:
235: 4.Swap:
236: 5.Search:
237: 6.display:
238: 7.exit:1
239:
240: enter the data21
241:
242: Enter your choice:
243: 1.insert:
244: 2.No of nodes in Longest path:

```

```
245: 3.Minimum value:
246: 4.Swap:
247: 5.Search:
248: 6.display:
249: 7.exit:3
250:
251: Minimum value = 21
252: Enter your choice:
253: 1.insert:
254: 2.No of nodes in Longest path:
255: 3.Minimum value:
256: 4.Swap:
257: 5.Search:
258: 6.display:
259: 7.exit:2
260:
261: No of nodes in Longest path=2
262: Enter your choice:
263: 1.insert:
264: 2.No of nodes in Longest path:
265: 3.Minimum value:
266: 4.Swap:
267: 5.Search:
268: 6.display:
269: 7.exit:6
270:
271: *56
272: *21
273: *567
274: Enter your choice:
275: 1.insert:
276: 2.No of nodes in Longest path:
277: 3.Minimum value:
278: 4.Swap:
279: 5.Search:
280: 6.display:
281: 7.exit:5
282:
283: enter data to be searched:21
284:
285: FOUND!!!
286: Enter your choice:
287: 1.insert:
288: 2.No of nodes in Longest path:
289: 3.Minimum value:
290: 4.Swap:
291: 5.Search:
292: 6.display:
293: 7.exit:4
294:
295: Enter your choice:
296: 1.insert:
297: 2.No of nodes in Longest path:
298: 3.Minimum value:
299: 4.Swap:
300: 5.Search:
301: 6.display:
302: 7.exit:6
303:
304: *56
305: *567
```

```
306: *21
307: Enter your choice:
308: 1.insert:
309: 2.No of nodes in Longest path:
310: 3.Minimum value:
311: 4.Swap:
312: 5.Search:
313: 6.display:
314: 7.exit:7
315:
316: -----
317: Process exited after 76.07 seconds with return value 0
318: Press any key to continue . . .
319: */
```