```
#include <stdio.h>
                                                            x->lchild=y;
#include <stdlib.h>
                                                            x=leftrotate(x);
                                                            return x;
                                                         }
typedef struct node{
  int data;
                                                         NODE* rotations(NODE* root){
  int bf;
                                                            if(root->bf==2&&root->lchild->bf==1){
  struct node *Ichild;
                                                               //printf("\nrightrotate\n");
                                                               root=rightrotate(root);
  struct node *rchild;
NODE:
                                                              return root:
                                                            if(root->bf==-2&& root->rchild->bf==-1){
int max(int a,int b){
  if(a>b)
                                                               //printf("\nleftrotate\n");
                                                               root=leftrotate(root);
     return a:
  return b;
                                                               return root:
}
                                                            if(root->bf==2&&root->lchild->bf==-1){
int height(NODE *root){
                                                               //printf("\nleftrightrotate\n");
  if(root==NULL)
                                                               root=leftrightrotate(root);
                                                              return root;
     return 0;
  int lheight=height(root->lchild);
                                                            if(root->bf==-2&&root->rchild->bf==1){
  int rheight=height(root->rchild);
  return max(lheight,rheight)+1;
                                                               //printf("\nrightleftrotate\n");
                                                               root=rightleftrotate(root);
}
                                                               return root;
void updateBF(NODE *root){
                                                            return root;
root->bf=height(root->lchild)-height(root->rchil
                                                         }
d);
}
                                                         NODE *delete(NODE *root,int dkey){
NODE* leftrotate(NODE* root){
                                                            if(root==NULL){
  NODE* pptr=root:
                                                               printf("key not prestent");
  NODE* aptr=pptr->rchild;
                                                               return root:
  pptr->rchild=aptr->lchild;
  aptr->lchild=pptr:
                                                            if(dkev<root->data){
                                                               root->lchild=delete(root->lchild,dkey);
  updateBF(pptr);updateBF(aptr);
  return aptr;
}
                                                            else if(dkey>root->data){
                                                               root->rchild=delete(root->rchild,dkey);
NODE* rightrotate(NODE* root){
  NODE* pptr=root;
NODE* aptr=pptr->lchild;
                                                            else{
  pptr->lchild=aptr->rchild;
                                                               if(root->lchild!=NULL &&
  aptr->rchild=pptr;
                                                         root->rchild!=NULL){
  updateBF(pptr);updateBF(aptr);
                                                                 NODE *suc=root->rchild;
  return aptr;
                                                                 while(suc->lchild!=NULL){
}
                                                                    suc=suc->lchild;
NODE* leftrightrotate(NODE *root){
                                                                 root->data=suc->data;
  NODE*x=root,*y=root->lchild;
                                                         root->rchild=delete(root->rchild,suc->data);
  y=leftrotate(y);
  x->lchild=y;
                                                               }
  x=rightrotate(x):
                                                               else{
  return x;
                                                                 NODE *temp=root;
                                                                 if(root->lchild!=NULL)
NODE* rightleftrotate(NODE *root){
                                                                    root=root->lchild;
  NODE*x=root,*y=root->lchild;
                                                                 else if(root->rchild!=NULL)
  y=rightrotate(y);
                                                                    root=root->rchild;
```

```
else
          root=NULL;
                                                          return;
                                                        }
       free(temp);
                                                        void display_inorder(NODE *root){
                                                          if(root==NULL)
     }
                                                             return;
                                                          display_inorder(root->lchild);
  }
     if(root!=NULL){
                                                          printf("%d:%d ",root->data,root->bf);
       updateBF(root);
                                                          display inorder(root->rchild);
     if(abs(root->bf)>1){
       printf("rotate\n");
                                                        void minmax(NODE*root){
       printf("%d\n",root->data);
                                                           NODE *tmp=root:
                                                          while(tmp->lchild!=NULL)
       root=rotations(root);
                                                             tmp=tmp->lchild;
                                                          printf("Min value: %d ",tmp->data);
  }
                                                          while(root->rchild!=NULL)
                                                             root=root->rchild;
  return root;
                                                           printf("Max value: %d\n",root->data);
}
NODE* insert(NODE *root,int data){
  if(root==NULL){
                                                        int width(NODE *root){
     root=(NODE*) malloc(sizeof(NODE));
                                                          if(root==NULL)
     root->data=data:
                                                             return 0:
     root->bf=0:
                                                          int lheight=height(root->lchild);
     root->lchild=NULL;
                                                          int rheight=height(root->rchild);
     root->rchild=NULL;
                                                          int lwidth=width(root->lchild);
                                                          int rwidth=width(root->rchild);
     return root;
                                                          return
  if(data<root->data){
                                                        max(lheight+rheight+1,max(lwidth,rwidth));
     root->lchild=insert(root->lchild,data);
  if(data>root->data){
                                                        void display decend(NODE *root){
     root->rchild=insert(root->rchild,data);
                                                           if(root==NULL)
                                                             return;
                                                          display decend(root->rchild);
                                                          printf("%d:%d ",root->data,root->bf);
  updateBF(root);
  if(abs(root->bf)>1){
                                                          display decend(root->lchild);
     root=rotations(root);
                                                        }
                                                        int main(){
                                                          NODE *root=NULL;
  return root;
}
                                                          int data;
                                                          int m=1;
void search(NODE *root,int data){
                                                          while(m){
  if(root==NULL){
                                                             printf("1:Insert a node\n2:Delete a
     printf("data not present\n");
                                                        node\n3:Display Inorder\n4:Search\n5:Max
     return;
                                                        and Min value\n6:Display in decending
                                                        order\n7: Width of the tree\n");
                                                             scanf("%d",&m);
  if(data<root->data){
     search(root->lchild,data);
                                                             switch(m){
                                                                case 1:
  else if(data>root->data){
                                                                  printf("Enter Data to be inserted\n");
     search(root->rchild,data);
                                                                  scanf("%d",&data);
                                                                  root=insert(root,data);
                                                                  break:
  }
  else{
                                                                case 2:
     printf("data present\n");
                                                                  printf("Enter Data to be deleted\n");
                                                                  scanf("%d",&data);
```

root=delete(root,data);	Enter data to be searched
break;	2
case 3:	data not present
<pre>printf("Inorder traverse: ");</pre>	1:Insert a node
display_inorder(root);	2:Delete a node
printf("\n");	3:Display Inorder
break;	4:Search
case 4:	5:Max and Min value
printf("Enter data to be searched\n");	6:Display in decending order
scanf("%d",&data);	7: Width of the tree
search(root,data);	1
break;	Enter Data to be inserted
•	5
case 5:	•
minmax(root);	1:Insert a node
break;	2:Delete a node
case 6:	3:Display Inorder
printf("Decending order\n");	4:Search
display_decend(root);	5:Max and Min value
printf("\n");	6:Display in decending order
break;	7: Width of the tree
•	
case 7:	1
<pre>printf("width is %d\n",width(root));</pre>	Enter Data to be inserted
break;	6
default:	1:Insert a node
break;	2:Delete a node
}	3:Display Inorder
}	4:Search
J	5:Max and Min value
1	
}	6:Display in decending order
	7: Width of the tree
O/P :	1
/tmp/dJnQTFLauf.o	Enter Data to be inserted
1:Insert a node	11
	1:Insert a node
2:Delete a node	1.113611 a 11006
2:Delete a node 3:Display Inorder	
3:Display Inorder	2:Delete a node
3:Display Inorder 4:Search	2:Delete a node 3:Display Inorder
3:Display Inorder 4:Search 5:Max and Min value	2:Delete a node 3:Display Inorder 4:Search
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value
3:Display Inorder 4:Search 5:Max and Min value	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order
3:Display Inorder4:Search5:Max and Min value6:Display in decending order7: Width of the tree1	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 3
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 3	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 3 Inorder traverse: 3:0 4:-1 5:0 6:1
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 3 1:Insert a node	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 3 Inorder traverse: 3:0 4:-1 5:0 6:1 1:Insert a node
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 3 1:Insert a node 2:Delete a node	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 3 Inorder traverse: 3:0 4:-1 5:0 6:1 1:Insert a node 2:Delete a node
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 3 1:Insert a node 2:Delete a node 3:Display Inorder	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 3 Inorder traverse: 3:0 4:-1 5:0 6:1 1:Insert a node 2:Delete a node 3:Display Inorder
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 3 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 3 Inorder traverse: 3:0 4:-1 5:0 6:1 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 3 1:Insert a node 2:Delete a node 3:Display Inorder	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 3 Inorder traverse: 3:0 4:-1 5:0 6:1 1:Insert a node 2:Delete a node 3:Display Inorder
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 3 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 3 Inorder traverse: 3:0 4:-1 5:0 6:1 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 3 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 3 Inorder traverse: 3:0 4:-1 5:0 6:1 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order
3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 4 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 1 Enter Data to be inserted 3 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value	2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 2 Enter Data to be deleted 11 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value 6:Display in decending order 7: Width of the tree 3 Inorder traverse: 3:0 4:-1 5:0 6:1 1:Insert a node 2:Delete a node 3:Display Inorder 4:Search 5:Max and Min value

Min value: 3 Max value: 6

1:Insert a node

2:Delete a node

3:Display Inorder

4:Search

5:Max and Min value

6:Display in decending order

7: Width of the tree

7

width is 4

1:Insert a node

2:Delete a node

3:Display Inorder

4:Search

5:Max and Min value

6:Display in decending order

7: Width of the tree

6

Decending order

6:1 5:0 4:-1 3:0

1:Insert a node

2:Delete a node

3:Display Inorder

4:Search

5:Max and Min value

6:Display in decending order

7: Width of the tree