```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>
#include<process.h>
#include<string.h>
struct node
{
  int info;
  struct node *llink;
  struct node *rlink;
};
typedef struct node* NODE;

NODE insert(int,NODE);
NODE getnode();
NODE minimum(NODE);
void inorder(NODE);
void preorder(NODE);
void postorder(NODE);
void iterative_search(int,NODE);
NODE delete_item(int,NODE);
void main()
{
  int ch,item,flag,item_deleted;
  NODE root=NULL,min;
  clrscr();
  for(;;)
  {
      printf("Press 1 .INSERT \n 2. Inorder\n 3 preorder\n");
      printf("4.postorder\n5:Search\n6:Find minimum\n7.delete\n");
      printf("Enter your choice\n");
      scanf("%d",&ch);
      switch(ch)
```

```
    {
case 1: printf("Enter the item to be inserted\n");
 scanf("%d",&item);
 root=insert(item,root);
 break;
case 2: printf("Inorder tree traversal\n");
 inorder(root);
 break;
case 3: printf("preorder tree traversal\n");
 preorder(root);
 break;
case 4: printf("postorder tree traversal\n");
 postorder(root);
 break;

case 5: if(root==NULL)
  printf("Tree is empty\n");
 else
  {
    printf("Enter the item to be search\n");
    scanf("%d",&item);
    iterative_search(item,root);
  }
 break;
 case 6: min=minimum(root);
  printf("smallest element =%d\n",min->info);
  break;
 case 7: printf("Enter the node to be deleted\n");
 scanf("%d",&item_deleted);
 root=delete_item(item,root);
 break;

/*      case 3: display(head);
```

```
        break; */
       default: exit(0);
}//switch
}//for
getch();
}//main
NODE getnode()
{
  NODE x;
  x=(NODE)malloc(sizeof(struct node));
  if(x==NULL)
  {
    printf("Out of memeory\n");
    exit(0);
  }
  return x;
}
void freenode(NODE temp)
{
  free(temp);
}
NODE minimum(NODE root)
{
 NODE cur;
 if(root==NULL) return root;
 cur=root;
 while(cur->llink!=NULL)
   cur=cur->llink;
 return cur;
}
NODE insert(int item,NODE root)
{
   NODE temp,cur,prev;
```

```
   temp=getnode();
   temp->info=item;
   temp->llink=temp->rlink=NULL;
   if(root==NULL) return temp;
   prev=NULL;
   cur=root;
   while( cur !=NULL)
   {
     prev=cur;
     if(item <cur->info)
       cur=cur->llink;
     else
       cur=cur->rlink;
   }
   if(item< prev->info)
     prev->llink=temp;
   else
     prev->rlink=temp;
return root;
}
void inorder(NODE root)
{
  if(root !=NULL)
  {
    inorder(root->llink);
    printf("%d  ",root->info);
    inorder(root->rlink);
  }
}
void preorder(NODE root)
{
  if(root !=NULL)
  {
```

```
    printf("%d  ",root->info);
    preorder(root->llink);
    preorder(root->rlink);
  }
}
void postorder(NODE root)
{
  if(root !=NULL)
  {
    postorder(root->llink);
    postorder(root->rlink);
    printf("%d  ",root->info);
  }
}
void iterative_search(int item,NODE root)
{
 while(root !=NULL && item != root->info)
 {
   if(item< root->info)
     root=root->llink;
   else
     root=root->rlink;
 }
 if(item==root->info)
   printf("Succesfull search\n");
 else
   printf("Unsuccesful Search\n");
// return root;
}
NODE delete_item(int item,NODE root)
{
  NODE parent,cur,suc,psuc,ptr;
  if(root->llink==NULL)
```

```
{
  printf("Tree is empty\n");
  return root;
}
parent=root;
cur=root->llink;
while(cur !=NULL && item !=cur->info)
{
  parent=cur;
  if(item<cur->info)
    cur=cur->llink;
  else
    cur=cur->rlink;
}
if(cur==NULL)
{
  printf("Item not found\n");
  return root;
}
if(cur->llink ==NULL)
  ptr=cur->rlink;
else if(cur->rlink ==NULL)
  ptr=cur->llink;
else
{
  psuc=cur;
  cur=cur->llink;
  while(suc->llink!=NULL)
  {
    psuc=cur;
    suc=suc->llink;
  }
  if(cur==psuc)
```

```
    {
       suc->llink=cur->rlink;
    }
    else
    {
       suc->llink=cur->llink;
       psuc->llink=suc->rlink;
       suc->rlink=cur->rlink;
    }
   ptr=suc;
}
if(parent->llink==cur)
    parent->llink=ptr;
else
   parent->rlink=ptr;
freenode(cur);
   return root;
}
```