BINARY TREE

```
#include<stdio.h>
#include<conio.h>
struct tnode
{
 struct tnode *lchild;
 int data;
 struct tnode *rchild;
};
typedef struct tnode tnode;
tnode *getnode();
main()
 {
 int a[100],i,n,item;
 tnode *root;
 clrscr();
 root=NULL;
 printf("Enter the no of elements:");
 scanf("%d",&n);
 printf("\nEnter the elements:");
 for(i=0;i<n;i++)
  scanf("%d",&a[i]);
 for(i=0;i<n;i++)
  Tinsert(&root,a[i]);
 printf("\nBinary tree:\n");
 Tdisplay(root,1);
 printf("\nInorder Traversal\n");
 Inorder(root);
 printf("\nPreorder Traversal\n");
 Preorder(root);
 printf("\nPostorder Traversal\n");
 Postorder(root);
 printf("\nEnter the element to delete:");
 scanf("%d",&item);
 Tdelete(&root,item);
 printf("\nBinary tree after deletion\n");
 Tdisplay(root,1);
 printf("\nEnter the element to search:");
 scanf("%d",&item);
 Tsearch(item);
 getch();
 }
Tinsert(tnode **rt,int item)
{
  tnode *current=(*rt),*temp;
  if((*rt)==NULL)
  {
   (*rt)=getnode();
   (*rt)->data=item;
   (*rt)->lchild=NULL;
   (*rt)->rchild=NULL;
```

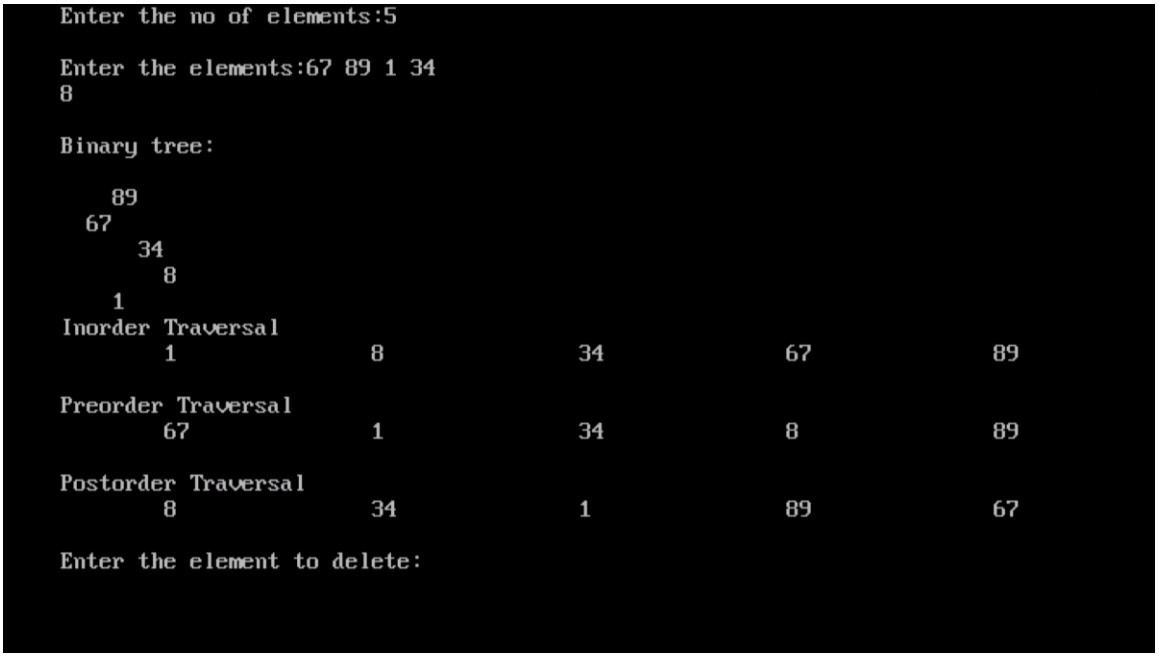
```
   return;
  }
  while(current!=NULL)
   {
   if(item<current->data)
   {
   if(current->lchild!=NULL)
     current=current->lchild;
   else
    {
    temp=getnode();
    current->lchild=temp;
    temp->data=item;
    temp->rchild=NULL;
    temp->lchild=NULL;
    return;
    }
    }
   else
   {
    if(item>current->data)
    {
     if(current->rchild!=NULL)
       current=current->rchild;
     else
     {
      temp=getnode();
      current->rchild=temp;
      temp->data=item;
      temp->rchild=NULL;
      temp->lchild=NULL;
      return;
      }
     }
     else
     {
     printf("\nWrong data");
     exit(0);
     }
     }
     }
     }
Inorder(tnode *rt)
{
 if(rt!=NULL)
 {
 Inorder(rt->lchild);
 printf("\t%d\t",rt->data);
 Inorder(rt->rchild);
 }
 else
  return;
```

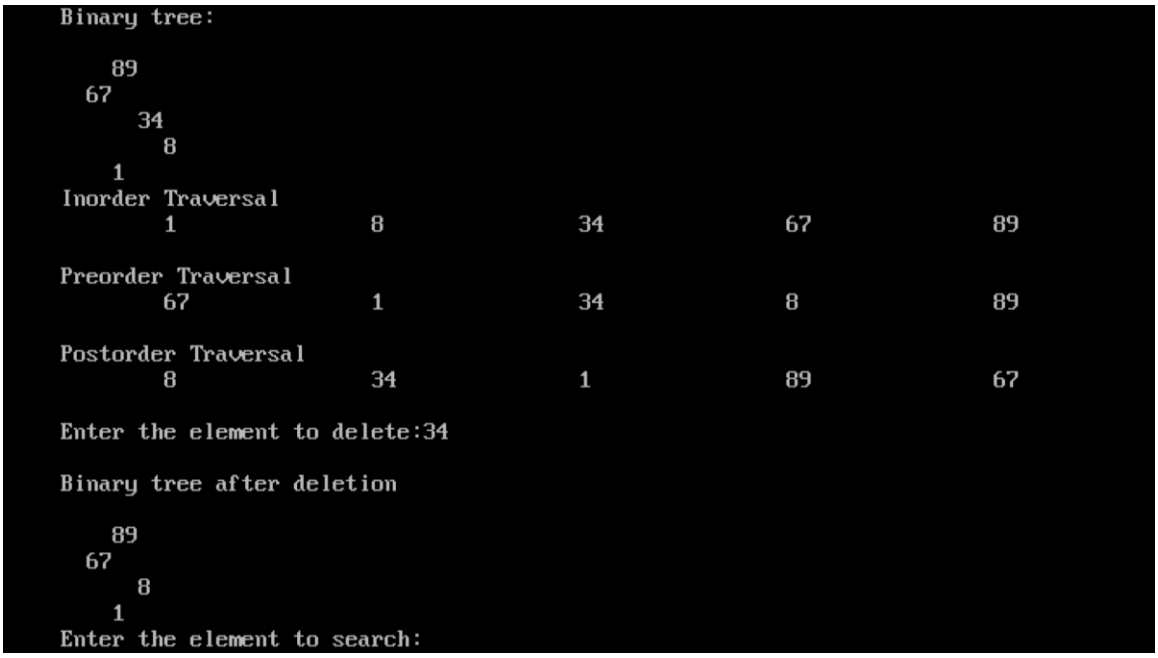
```
}
Preorder(tnode *rt)
{
 if(rt!=NULL)
 {
 printf("\t%d\t",rt->data);
 Preorder(rt->lchild);
 Preorder(rt->rchild);
 }
 else
  return;
}
Postorder(tnode *rt)
{
 if(rt!=NULL)
 {
  Postorder(rt->lchild);
  Postorder(rt->rchild);
  printf("\t%d\t",rt->data);
 }
 else
  return;
}
Tdisplay(tnode *rt,int level)
{
 int i;
 if((rt)!=NULL)
 {
  Tdisplay((rt)->rchild,level+1);
  printf("\n");
  for(i=0;i<level;i++)
   printf("  ");
  printf("%d",(rt)->data);
  Tdisplay((rt)->lchild,level+1);
  }
}
Tdelete(tnode **rt,int item)
{
 tnode *current;
 if(*rt==NULL)
  {
  printf("\nError");
  getch();
  return;
  }
 if(item<(*rt)->data)
   Tdelete(&((*rt)->lchild),item);
 else
 {
  if(item>(*rt)->data)
    Tdelete(&((*rt)->rchild),item);
  else
```

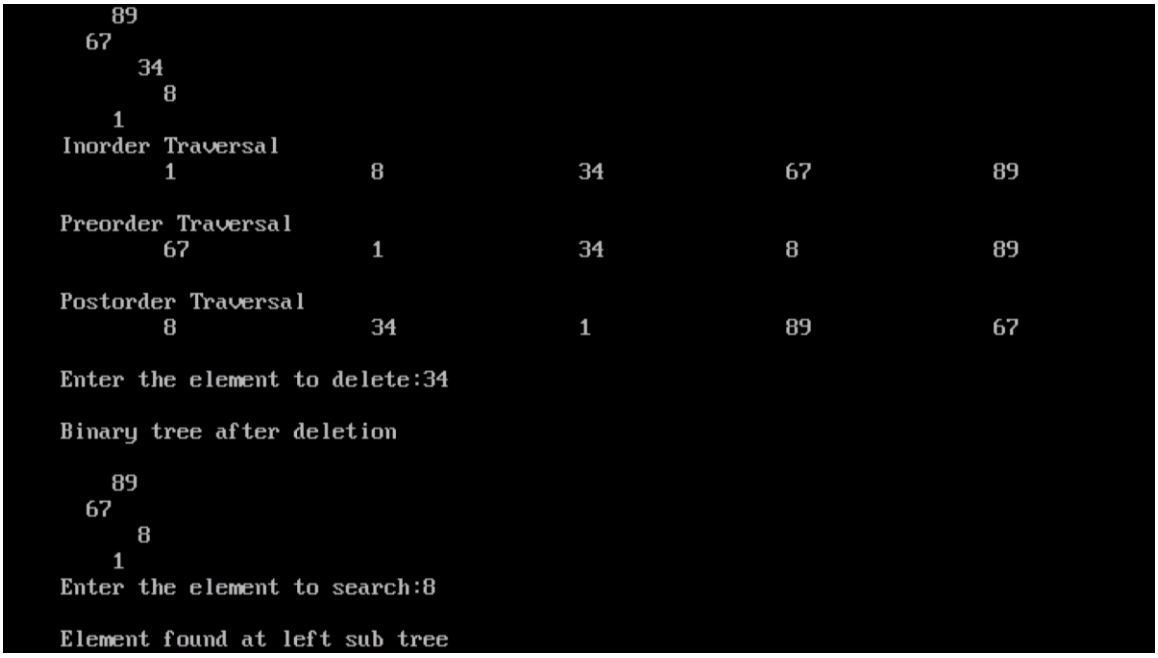
```
    {
    current=(*rt);
    if(current->rchild==NULL)
    {
    (*rt)=(*rt)->lchild;
    free(current);
    }
   else
   {
   if(current->lchild==NULL)
   {
   (*rt)=(*rt)->rchild;
   free(current);
   }
   else
   {
   current=(*rt)->rchild;
    while(current->lchild!=NULL)
     current=current->lchild;
    current->lchild=(*rt)->lchild;
    current=(*rt);
    (*rt)=(*rt)->rchild;
    free(current);
    }
    }
   }
   }
}
Tsearch(int item)
{
 tnode *temp,*root;
if(root==NULL)
 {
  printf("\nTree is empty");
  return;
 }
 if(item==root->data)
  {
  printf("\nElement found at root");
  return;
  }
 else if(item<root->data)
 {
  printf("\nElement found at left sub tree");
  return;
  }
 else
 {
  printf("\nElement found at right sub tree");
  return;
  }
}
```

```
tnode *getnode()
{
 tnode *p;
 p=(tnode *)malloc(sizeof(tnode));
 return(p);
}
freenode(tnode *p)
 {
 free(p);
 }
```

OUTPUT




Binary tree:
89
67
34
8
1
Inorder Traversal
1 8 34 67 89
Preorder Traversal
67 1 34 8 89
Postorder Traversal
8 34 1 89 67
Enter the element to delete:34
Binary tree after deletion
89
67
8
1
Enter the element to search:


89
67
34
8
1
Inorder Traversal
1 8 34 67 89
Preorder Traversal
67 1 34 8 89
Postorder Traversal
8 34 1 89 67
Enter the element to delete:34
Binary tree after deletion
89
67
8
1
Enter the element to search:8
Element found at left sub tree