```c
#include <stdio.h>
#include <stdlib.h>
typedef struct node
{
int info;
struct node *left,*right;
}node;
//-----------------------
node* createnode(int item)
{
node *temp;
temp=(node*)malloc(sizeof(temp));
if(temp==NULL)
printf("\nMemory insufficient");
temp->info=item;
temp->left=temp->right=NULL;
return temp;
}
//-----------------------------
void inorder(node *ROOT)
{
if(ROOT)
{
inorder(ROOT->left);
printf("%d  ",ROOT->info);
inorder(ROOT->right);
}
}
//-------------------------------
void preorder(node *ROOT)
{
if(ROOT)
{
printf("%d  ",ROOT->info);
preorder(ROOT->left);
preorder(ROOT->right);
}
}
//------------------------------
void postorder(node *ROOT)
{
if(ROOT)
{
postorder(ROOT->left);
postorder(ROOT->right);
printf("%d  ",ROOT->info);
}
}
//--------------------------------
node* bst_insert(node *ROOT,node *temp)
{
if(ROOT==NULL)
   ROOT=temp;
   else if(ROOT->info==temp->info)
```

```
       printf("\n Duplicate node value");
       else if(ROOT->info > temp->info)
                ROOT->left=bst_insert(ROOT->left,temp);
            else
                  ROOT->right=bst_insert(ROOT->right,temp);

return ROOT;
}
//-----------------------------------------------------------
node* bst_insert1(node *ROOT,node *temp)
{       node *p;
if(ROOT==NULL)
  ROOT=temp;
   else if(ROOT->info==temp->info)
       printf("\n Duplicate node value");
       else
       {
       p=ROOT;
       while(p)
       {
               if(p->info > temp->info)
               {     if(p->left)
                         p=p->left;
                     else
                      {
                         break;
                      }
               }
               else
               {   if(p->right)
                     p=p->right;
                   else
                   {
                      break;
                   }
               }
       }// end of while
       if(p->left==NULL)
           p->left=temp;
       else
           p->right=temp;

       }// end of else

return ROOT;
}
//-----------------------------------------------------------
void bst_search(node *ROOT,int val)
{
 if(ROOT==NULL)
    printf("\n search item not present");
 else if(ROOT->info==val)
       printf("\n item found");
       else if(ROOT->info>val)
```

```
               bst_search(ROOT->left,val);
            else
              bst_search(ROOT->right,val);
}
//-------------------------------------------
/*node* bst_delete(node*ROOT,int val,node *p)
{  node *T,*prev;

if(ROOT==NULL)
     printf("\n Not found");
  else if(ROOT->info > val)
          bst_delete(ROOT->left,val,ROOT);
       else if(ROOT->info < val)
               bst_delete(ROOT->right,val,ROOT);
            else
            {
             //----------case 1--------
               if(ROOT->left==NULL&&ROOT->right==NULL)
                  if(p->left==ROOT)
                     p->left=NULL;
                  else
                     p->right=NULL;
             //-------------case 2---------------
               else if(ROOT->left==NULL)
                      if(p->left==ROOT)
                         p->left=ROOT->right;
                      else
                         p->right=ROOT->right;
             //-------------case 3---------------
               else if(ROOT->right==NULL)
                      if(p->left==ROOT)
                         p->left=ROOT->left;
                      else
                         p->right=ROOT->left;
            //----------------case 4----------------------
               else
               {
                T=ROOT->right;
                prev=ROOT;
                while(T->left)
                {     prev=T;
                      T=T->left;
                 }
                 ROOT->info=T->info;  // copy
                 if(T->right!=NULL)
                    prev->left=T->right;
                 else
                    prev->left=NULL;
                 free(T);
              }
              }

} */
//================================
int main()
{
int i,n,val,item;
node *ROOT,*temp;
ROOT=NULL;

while(1)
{
printf("\n1.BST INSERT \n2.BST SEARCH \n3.BST DELETE \n4.BST TRAVERSAL
\n5.EXIT");
scanf("%d",&n);
switch(n)
{
case 1:  printf("\n enter item to insert into bst");
         scanf("%d",&item);
         temp=createnode(item);
         ROOT=bst_insert1(ROOT,temp);
         inorder(ROOT);
         break;
case 2:  printf("\n enter item to search in bst");
         scanf("%d",&val);
         bst_search(ROOT,val);
         break;
//case 3:  printf("\n enter item to delete from bst");
  //       scanf("%d",&val);
    //     bst_delete(ROOT,val,temp);
      //   break;
case 4:  printf("\n PREORDER=>");
         system("PAUSE");
         preorder(ROOT);
         printf("\n INEORDER=>");
         inorder(ROOT);
         printf("\n POSTEORDER=>");
         postorder(ROOT);
         system("PAUSE");
         break;
default: exit(0);
}
}

      system("PAUSE");
      return 0;
}
```