7) Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers.

a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2
b. Traverse the BST in In-order, preorder, post-Order
c. Search the BST for a given element (KEY) and report the appropriate message
d. Display the height of binary trees
e. Exit

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
 int item;
 struct node *llink, *rlink;
}NODE;
```

NODE* getnode()

```
{
 NODE* x;
 x=(NODE*)malloc(sizeof(NODE));
 scanf("%d",&x->item);
 x->llink=x->rlink=NULL;
 return x;
}
```

NODE* insert(NODE* root)

```
{
 NODE *temp,*cur,*prev;
 temp=getnode();
 if(root==NULL)
 {
        root=temp;
 }
 else
 {
        prev=NULL;
         cur=root;
        while(cur!=NULL)
        {
                prev=cur;
                if(temp->item>cur->item)
                cur=cur->rlink;
                else
                cur=cur->llink;
        }
        if(temp->item>prev->item)
        prev->rlink=temp;
        else
         prev->llink=temp;
```

```
 }
return root;
}

void search(NODE *root)
{
 int item;
 NODE *cur;
 cur=root;
 if(root==NULL)
 {
        printf("Tree is empty\n");
 }
 else
 {
        printf("Enter the item to be searched: ");
        scanf("%d",&item);
        while(cur!=NULL)
        {
                if(cur->item==item)
                 break;
                if(cur->item<item)
                cur=cur->rlink;
                else
                cur=cur->llink;
         }
         if(cur!=NULL)
         {
                printf("Item found\n");
        }
        else
        {
                printf("Item Not found");
        }
 }
}

void preorder(NODE *root)
{
 if(root==NULL) return;
 printf("%d\t",root->item);
 preorder(root->llink);
 preorder(root->rlink);
}

void postorder(NODE *root)
{
 if(root==NULL) return;
 postorder(root->llink);
```

```
 postorder(root->rlink);
 printf("%d\t",root->item);
}

void inorder(NODE *root)
{
 if(root==NULL) return;
 inorder(root->llink);
 printf("%d\t",root->item);
 inorder(root->rlink);
}
int find_height(NODE *root)
{
   if (root==NULL)
   {
    return -1;
   }
   else
   {
      int lheight = find_height(root->llink);
      int rheight = find_height(root->rlink);
      if (lheight > rheight)
          return(lheight + 1);
      else
          return(rheight + 1);
   }
}
int main()
{
 int ch,i,n,ht;
 NODE *root=NULL;
 while(1)
 {
  printf("\n 1.Create\t 2.Traverse\t 3.Search\t 4.Height\t 5.Exit\n");
  printf("Enter your choice: ");
  scanf("%d",&ch);
   switch(ch)
   {
    case 1:printf("Enter the number of nodes to be inserted: ");
          scanf("%d",&n);
          printf("Enter the tree nodes\n");
          for(i=0;i<n;i++)
          {
           root=insert(root);
          }
          break;
    case 2:printf("\n Preorder Traversal: ");
          preorder(root);
          printf("\n Inorder Traversal: ");
```

```
            inorder(root);
            printf("\n Postorder Traversal: ");
            postorder(root);
            break;
       case 3:search(root);
            break;
       case 4:ht=find_height(root);
            printf("\n Height of the tree = %d\n",ht);
            break;
       case 5:exit(0);
       default:printf("\n Invalid Choice\n");
             break;
     }
   }
   return 0;
}
```