**10)**

**Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2**

**b. Traverse the BST in Inorder, Preorder and Post Order**

**c. Search the BST for a given element (KEY) and report the appropriate message**

**e. Exit**

#include <stdio.h>

#include<stdlib.h>

typedef struct NODE

{

int info;

struct NODE \*lchild;

struct NODE \*rchild;

}node;

node \*root=NULL;

node \*tree;

void create();

node\* insert(int);

void preorder(node\*);

void inorder(node\*);

void postorder(node\*);

int search(node \*,int);

int main()

{

int ch,key,flag=0;

do

{

printf("\n1.create\n2.preorder\n3.inorder\n4.postorder\n5.search\n6.exit\n");

printf("enter your choice:");

scanf("%d",&ch);

switch(ch)

{

case 1:create();

break;

case 2:preorder(root);

break;

case 3:inorder(root);

break;

case 4:postorder(root);

break;

case 5:printf("enter the search key:");

scanf("%d",&key);

flag=search(root,key);

if(flag==1)

printf("element is found");

else

printf("not found");

break;

case 6:exit(0);

default:printf("invalid choice");

}

}while(ch!=6);

return 0;

}

void create()

{

int i,n,e;

printf("enter the no of nodes:");

scanf("%d",&n);

printf("enter the element:");

for(i=0;i<n;i++)

{

scanf("%d",&e);

root=insert(e);

}

printf("tree constructed\n");

}

node\* insert(int e)

{

node \*nn,\*prev,\*temp;

nn=(node\*)malloc(sizeof(node));

nn->rchild=NULL;

nn->lchild=NULL;

nn->info=e;

if(root==NULL)

{

root=nn;

return(root);

}

else

{

temp=root;

while(temp!=NULL)

{

prev=temp;

if(temp->info>nn->info)

temp=temp->lchild;

else if(temp->info<nn->info)

temp=temp->rchild;

else

{

printf("duplicate");

return(root);

}

}

if(prev->info<nn->info)

prev->rchild=nn;

else

prev->lchild=nn;

return(root);

}

}

void preorder(node \*tree)

{

if(tree==NULL)

return;

else

{

printf("%d",tree->info);

preorder(tree->lchild);

preorder(tree->rchild);

}

}

void postorder(node \*tree)

{

if(tree==NULL)

return;

else

{

postorder(tree->lchild);

postorder(tree->rchild);

printf("%d",tree->info);

}

}

void inorder(node \*tree)

{

if(tree==NULL)

return;

else

{

inorder(tree->lchild);

printf("%d",tree->info);

inorder(tree->rchild);

}

}

int search(node \*root,int e)

{

if(root==NULL)

{

return 0;

}

else if(root->info==e)

return 1;

else if(root->info>e)

search(root->lchild,e);

else

search(root->rchild,e);

return 0;

}