****

**ABDUL MUTEEN MASOOD**

**SP20-BSE-085**

**SECTION:A**

# include <stdio.h>

# include <malloc.h>

struct node

{

int info;

struct node \*left;

struct node \*right;

}\*root;

void search(int item, struct node \*par, struct node \*loc);

void insert(int item);

int deletion(int item);

int preorder(struct node \*ptr);

void inorder(struct node \*ptr);

void postorder(struct node \*ptr);

void display(struct node \*ptr, int level);

void search(int item, struct node \*\*par, struct node \*\*loc)

{

struct node \*ptr, \*ptrsave;

if(root == NULL)

{

\*loc = NULL;

\*par = NULL;

return;

}

if(item == root->info)

{

\*loc = root;

\*par = NULL;

return;

}

if(item < root->info)

ptr = root->left;

else

ptr = root->right;

ptrsave = root;

while(ptr != NULL)

{

if(item == ptr->info)

{

\*loc = ptr;

\*par = ptrsave;

return;

}

ptrsave = ptr;

if(item < ptr->info)

ptr = ptr->left;

else

ptr = ptr->right;

}

\*loc = NULL;

\*par = ptrsave;

}

void insert(int item)

{ struct node \*tmp, \* parent, \*location;

search(item, &parent, &location);

if(location != NULL)

{

printf("Item already present");

return;

}

tmp = (struct node \*)malloc(sizeof(struct node));

tmp->info = item;

tmp->left = NULL;

tmp->right = NULL;

if(parent == NULL)

root = tmp;

else

if(item < parent->info)

parent->left = tmp;

else

parent->right = tmp;

}

void firstOption(struct node \*par, struct node \*loc )

{

if(par == NULL)

root = NULL;

else

if(loc == par->left)

par->left = NULL;

else

par->right = NULL;

}

void secondOption(struct node \*par, struct node \*loc)

{

struct node \*child;

if(loc->left != NULL)

child = loc->left;

else

child = loc->right;

if(par == NULL )

root = child;

else

if( loc == par->left)

par->left = child;

else

par->right = child;

}

void thirdOption(struct node \*par, struct node \*loc)

{

struct node \*ptr, \*ptrsave, \*suc,\*parsuc;

ptrsave = loc;

ptr = loc->right;

while(ptr->left != NULL)

{

ptrsave = ptr;

ptr = ptr->left;

}

suc = ptr;

parsuc = ptrsave;

if(suc->left == NULL && suc->right == NULL)

firstOption(parsuc, suc);

else

secondOption(parsuc, suc);

if(par == NULL)

root = suc;

else

if(loc == par->left)

par->left = suc;

else

par->right = suc;

suc->left = loc->left;

suc->right = loc->right;

}

int deletion(int item)

{

struct node \*parent, \*location;

if(root == NULL)

{

printf("Tree empty");

return 0;

}

search(item, &parent, &location);

if(location == NULL)

{

printf("Item not present in tree\n");

return 0;

}

if(location->left == NULL && location->right == NULL)

firstOption(parent, location);

if(location->left != NULL && location->right == NULL)

secondOption(parent, location);

if(location->left == NULL && location->right != NULL)

secondOption(parent, location);

if(location->left != NULL && location->right != NULL)

thirdOption(parent, location);

free(location);

}

int preorder(struct node \*ptr)

{

if(root == NULL)

{

printf("Tree is empty\n");

return 0;

}

if(ptr != NULL)

{

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

preorder(ptr->left);

preorder(ptr->right);

}

}

void inorder(struct node \*ptr)

{

if(root == NULL)

{

printf("Tree is empty\n");

return;

}

if(ptr != NULL)

{

inorder(ptr->left);

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

inorder(ptr->right);

}

}

void postorder(struct node \*ptr)

{

if(root == NULL)

{

printf("Tree is empty\n");

return;

}

if(ptr != NULL)

{

postorder(ptr->left);

postorder(ptr->right);

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

}

}

void display(struct node \*ptr,int level)

{

int i;

if ( ptr != NULL )

{

display(ptr->right, level+1);

printf("\n");

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

printf(" ");

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

display(ptr->left, level+1);

}

}

main()

{

int option, num;

root = NULL;

do

{

printf("\n");

printf("\t\t\t\t\t........Option Menu........\n\n\n");

printf("\t\t\t\t\t-------------------------\n");

printf("\t\t\t\t\t| 1.Insert\t\t|\n");

printf("\t\t\t\t\t| 2.Delete\t\t|\n");

printf("\t\t\t\t\t| 3.Inorder Traversal\t|\n");

printf("\t\t\t\t\t| 4.Preorder Traversal\t|\n");

printf("\t\t\t\t\t| 5.Postorder Traversal\t|\n");

printf("\t\t\t\t\t| 6.Display\t\t|\n");

printf("\t\t\t\t\t| 7.Quit\t\t|\n");

printf("\t\t\t\t\t-------------------------\n");

printf("Enter your option : ");

scanf("%d", &option);

switch(option)

{

case 1:

printf("Enter the number to be inserted : ");

scanf("%d", &num);

insert(num);

break;

case 2:

printf("Enter the number to be deleted : ");

scanf("%d", &num);

deletion(num);

break;

case 3:

inorder(root);

break;

case 4:

preorder(root);

break;

case 5:

postorder(root);

break;

case 6:

display(root,1);

break;

case 7:

break;

default:

printf("Wrong option\n");

}

}while(option != 7);

}