**1) write a C Program to create a tree, traverse and search for an item.**

#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<alloc.h>

#include<process.h>

struct tree

{

int info;

struct tree \*lchild;

struct tree \*rchild;

};

typedef struct tree \*TREE;

TREE getnode()

{

TREE x;

x=(TREE) malloc(sizeof(struct tree));

if(x==NULL)

{

printf("\nNO MEMORY!...!");

getch();

exit(1);

}

return x;

}

TREE insert(int item,TREE root)

{

TREE temp,cur,prev;

char path[20];

int i,len;

temp=getnode();

temp->info=item;

temp->lchild=temp->rchild=NULL;

if(root==NULL)

return temp;

printf("\nEnter the path of insertion : ");

fflush(stdin);

gets(path);

len=strlen(path);

prev=NULL;

cur=root;

for(i=0;i<len&&cur!=NULL;i++)

{

prev=cur;

if(path[i]=='l'||path[i]=='L')

cur=cur->lchild;

else if(path[i]=='r'||path[i]=='R')

cur=cur->rchild;

else

{

printf("\nInvalid path...!!");

free(temp);

return root;

}

}

if(cur!=NULL||i!=len)

{

printf("\nInsertion not possible !...!");

free (temp);

return root;

}

if(path[--i]=='l')

prev->lchild=temp;

else

prev->rchild=temp;

return root;

}

void preorder(TREE root)

{

if(root!=NULL)

{

printf("\t%d",root->info);

preorder(root->lchild);

preorder(root->rchild);

}

}

void inorder(TREE root)

{

if(root!=NULL)

{

inorder(root->lchild);

printf("\t%d",root->info);

inorder(root->rchild);

}

}

void postorder(TREE root)

{

if(root!=NULL)

{

postorder(root->lchild);

postorder(root->rchild);

printf("\t%d",root->info);

}

}

void search(int item,TREE root, int \*flag)

{

if(root!=NULL)

{

search(item,root->lchild,flag);

if(item==root->info)

{

\*flag=1;

return;

}

search(item,root->rchild,flag);

}

}

void main()

{

TREE root=NULL;

int choice,item,flag,k=0;

while(k!=1)

{

clrscr();

printf("\n\n\n\t\t\t\*\*\*\*\*\*MAIN MENU\*\*\*\*\*\*\*");

printf("\n\n\n\tENTER 1 --> TO INSERT INTO THE TREE");

printf("\n\tENTER 2 --> TO TRAVERSE THE TREE IN PREORDER");

printf("\n\tENTER 3 --> TO TRAVERSE THE TREE IN INORDER");

printf("\n\tENTER 4 --> TO TRAVERSE THE TREE IN POSTORDER");

printf("\n\tENTER 5 --> TO SEARCH FOR AN ITEM IN THE TREE");

printf("\n\tENTER 6 --> TO QUIT");

printf("\n\n\nEnter your choice : ");

scanf("%d",&choice);

switch(choice)

{

case 1:

printf("\nEnter the item to be inserted : ");

scanf("%d",&item);

root=insert(item,root);

break;

case 2:

if(root==NULL)

printf("\nTree is empty");

else

{

printf("\nPreorder traversal is : \n");

preorder(root);

}

break;

case 3:

if(root==NULL)

printf("\nTree is empty");

else

{

printf("\nInorder traversal is : \n");

inorder(root);

}

break;

case 4:

if(root==NULL)

printf("\nTree is empty");

else

{

printf("\nPostorder traversal is : \n");

postorder(root);

}

break;

case 5:

if(root==NULL)

printf("\nTree is empty");

else

{

printf("\nEnter item to be searched : ");

scanf("%d",&item);

flag=0;

search(item,root,&flag);

if(flag==1)

printf("\nSearch successful");

else

printf("\nSearch Unsucessful");

}

break;

case 6:

k=1;

printf("\nTHANK YOU.. HAVE A NICE DAY QUITTING FROM THE PROGRAM!!!");

break;

}

getch();

}

}