//Bank Management System

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

struct node

{

int acc\_no565;

int ph\_no;

int amt;

char acc\_name[100];

char gender[10];

struct node \*left;

struct node \*right;

}\*root=NULL;

main()

{

int item,x,ch;

do

{

printf("\n\n");

printf("\t\t\t\tMAIN MENU\n");

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

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

printf("\t\t1.New Account Form\n");

printf("\t\t2.Traverse\n");

printf("\t\t3.Delete\n");

printf("\t\t4.Quit\n");

printf("\t\t");

scanf("%d",&x);

switch(x)

{

case 1:

printf("Enter Account Number:\n");

scanf("\t\t%d",&item);

insert(item);

break;

case 2:

traverse(root);

break;

case 3:

printf("\t\tEnter Number to be deleted:\n");

scanf("%d",&item);

delete(item);

break;

case 4:

exit(1);

default:

printf("\nWrong Choice! Try Again\n");

}

}while(ch!=4);

}

struct node \*getnode()

{

struct node \*p;

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

p->right=NULL;

p->left=NULL;

return(p);

}

/////////////////////////////////////////////////

/////////////// Insert Function ////////////////

///////////////////////////////////////////////

insert(int item)

{

struct node \*new,\*loc,\*par;

search(item,&par,&loc);

if(loc!=NULL)

{

printf("Duplicate Value.Therefore can't insert");

}

else

{

new=getnode();

new->acc\_no=item;

}

if(par==NULL)

{

root=new;

return;

}

else if(item>par->acc\_no)

{

par->right=new;

}

else

{

par->left=new;

}

}

/////////////////////////////////////////////////

/////////////// Search Function ////////////////

///////////////////////////////////////////////

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

{

struct node \*ptr,\*save;

if(root==NULL)

{

\*loc=NULL;

\*par=NULL;

return;

}//If tree is empty

else if(item==root->acc\_no)

{

\*loc=root;

\*par=NULL;

return;

}//Tree has only one node

if(item<root->acc\_no)

{

ptr=root->left;

save=root;

}//Item is smaller than root

else

{

ptr=root->right;

save=root;

}//Item is greater than root

while(ptr!=NULL)

{

if(item==ptr->acc\_no)

{

\*loc=ptr;

\*par=save;

return;

}//If Item is found on ptr

else if(item>ptr->acc\_no)

{

save=ptr;

ptr=ptr->right;

}//If Item is bigger than ptr

else

{

save=ptr;

ptr=ptr->left;

}//If Item is smaller than ptr

}

\*par=save;

\*loc=NULL;

}

traverse(struct node \*root)

{

if(root==NULL)

return;

traverse(root->left);

printf("%d\n",root->acc\_no);

traverse(root->right);

}

//--------------------------------------------//

// Delete Function //

//------------------------------------------//

delete(int item)

{

struct node \*par,\*loc;

search(item,&par,&loc);

if(loc==NULL)

{

printf("\t\tThe Node does not exist\n");

}

else if(loc->left!=NULL&&loc->right!=NULL)

{

caseb(&par,&loc);

}

else

{

casea(&par,&loc);

printf("\t\tThe Node is deleted successfully....");

free(loc);

}

}

//---------------------------------------------//

// When node contains either 1 or 0 child //

//-------------------------------------------//

casea(struct node \*\*par,struct node \*\*loc)

{

struct node \*child;

if((\*loc)->left==NULL&&(\*loc)->right==NULL)

child=NULL;//No child node

else if((\*loc)->left!=NULL)

child=(\*loc)->left;//child is in left subtree

else

child=(\*loc)->right;//child is in right subtree

if(\*par!=NULL)

{

if((\*loc)==(\*par)->left)

{

(\*par)->left=child;

}

else

{

(\*par)->right=child;

}

}

else

root=child;//Tree contains only one node

}

////////////////////////////////////////

//// When node contains 2 children ////

//////////////////////////////////////

caseb(struct node \*\*par,struct node \*\*loc)

{

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

ptr=(\*loc)->right;

save=\*loc;

while(ptr->left!=NULL)

{

save=ptr;

ptr=ptr->left;

}

suc=ptr;

parsuc=save;

casea(&parsuc,&suc);

if((\*par)!=NULL)

{

if((\*par)->left==\*loc)

(\*par)->left=suc;

else

(\*par)->right=suc;

}

root=suc;

suc->left=(\*loc)->left;

suc->right=(\*loc)->right;

printf("The node deleted is caseb = %d",suc);

}