#include<iostream>

using namespace std;

class Node

{

public:

int key;

Node \*ln, \*rn;

};

class Tree

{

public:

Node\* root;

Node\* createTree(int key)

{

root = new Node();

root->key = key;

root->ln = NULL;

root->rn = NULL;

return root;

}

void insertNode(int key, Node\* root)

{

Node\* node = new Node();

node->key = key;

if (root->key >key) //COMPARISION

{

if (root->ln == NULL)

{

root->ln = node;

}

else

insertNode(key, root->ln);

}

else if (root->key < key)

{

if (root->rn == NULL)

{

root->rn = node;

}

else

insertNode(key, root->rn);

}

else

cout<<"No duplicate keys are allowed"<<endl;

}

void searchNode(int searchkey, Node\* root)

{

if(root == NULL)

cout<<"No tree present";

if(root->key==searchkey)

{

cout<<"Key found !!!"<<endl;

}

else if (root->key > searchkey)

{

if (root->ln == NULL)

{

cout<<"Key is not present in the tree"<<endl;

}

else

searchNode(searchkey, root->ln);

}

else if (root->key < searchkey)

{

if (root->rn == NULL)

{

cout<<"Key is not present in the tree"<<endl;

}

else

searchNode(searchkey, root->rn);

}

}

void displayInorder(Node\* root)

{

if (root != NULL)

{

displayInorder(root->ln);

cout << root->key << endl;

displayInorder(root->rn);

}

}

void displaymin(Node\* root)

{

while (root->ln != NULL)

{

root = root->ln;

}

cout<<"Minimum number is " << root->key <<endl;

}

void displaymax(Node\* root){

while(root->rn != NULL){

root = root->rn;

}

cout<<"Maximum number is " << root->key <<endl;

}

int longestPath(Node\* root)

{

if(root==NULL)

return 0;

int Lctr = longestPath(root->ln);

int Rctr = longestPath(root->rn);

if(Lctr>Rctr)

return (Lctr+1);

else return (Rctr+1);

}

Node\* swapNodes(Node\* root)

{

Node\* temp;

if(root==NULL)

return NULL;

temp = root->ln; //SWAPPING

root->ln=root->rn;

root->rn=temp;

swapNodes(root->ln);

swapNodes(root->rn);

}

};

int main()

{

int choice, order, flag = 0;

int key, searchKey;

Tree t1;

Node\* root;

do

{

cout<<" MENU "<<endl;

cout<<"1. Insert Node "<<endl;

cout<<"2. Display Inorder of the Tree"<<endl;

cout<<"3. Display Min"<<endl;

cout<<"4. Display Max"<<endl;

cout<<"5. Swap left and right subtrees"<<endl;

cout<<"6. Search in tree"<<endl;

cout<<"7. Number of nodes in the longest path"<<endl;

cout<<"8. EXIT "<<endl;

cout<<"Enter choice: ";

cin >> choice;

switch (choice)

{

case 1:

cout << "\nEnter the number ";

cin >> key;

if (flag == 0)

{

root = t1.createTree(key);

flag = 1;

}

else

{

t1.insertNode(key, root);

}

break;

case 2:

t1.displayInorder(root);

break;

case 3:

t1.displaymin(root);

break;

case 4:

t1.displaymax(root);

break;

case 5:

t1.swapNodes(root);

cout<<"Swapped! The new list is : ";

t1.displayInorder(root);

break;

case 6:

cout << "\nEnter the key you want to search: ";

cin >> searchKey;

t1.searchNode(searchKey,root);

break;

case 7:

cout<<"Number of nodes in the longest path is : "<<t1.longestPath(root);

break;

case 8:

exit(0);

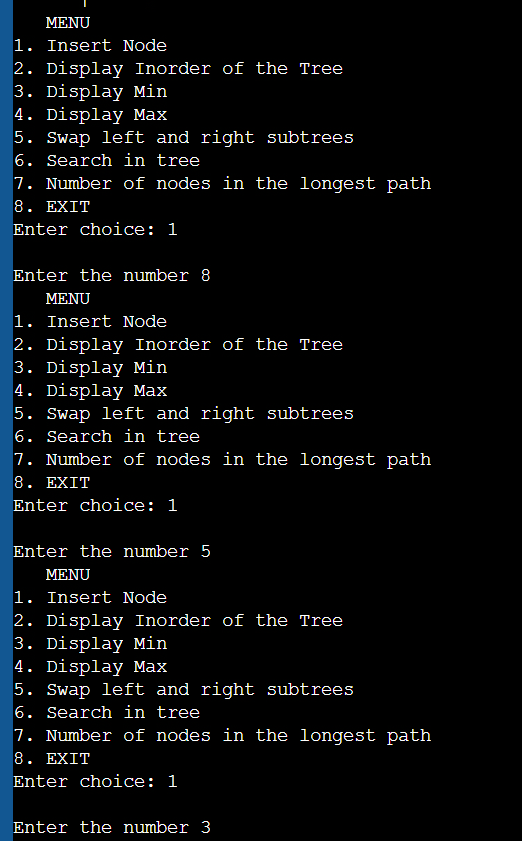
}

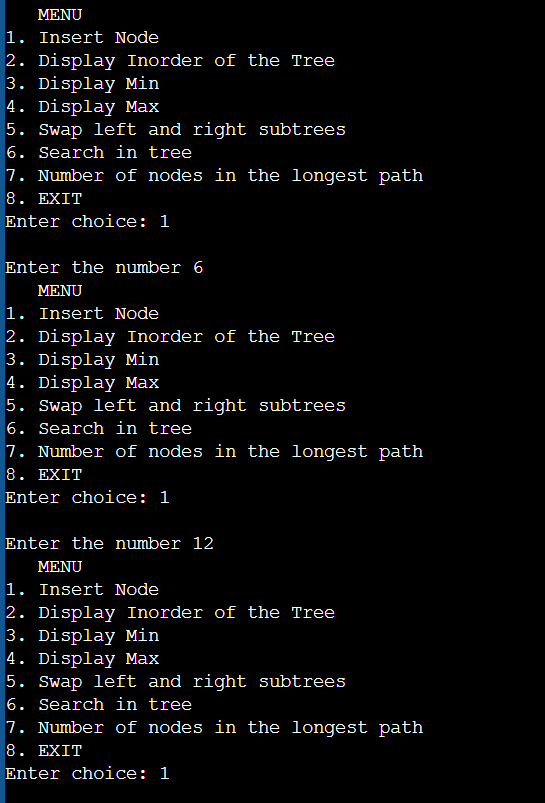
}

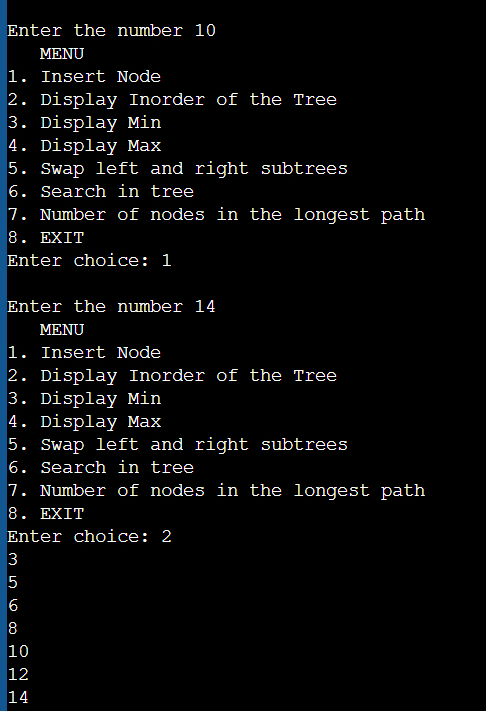
while (choice != 8);

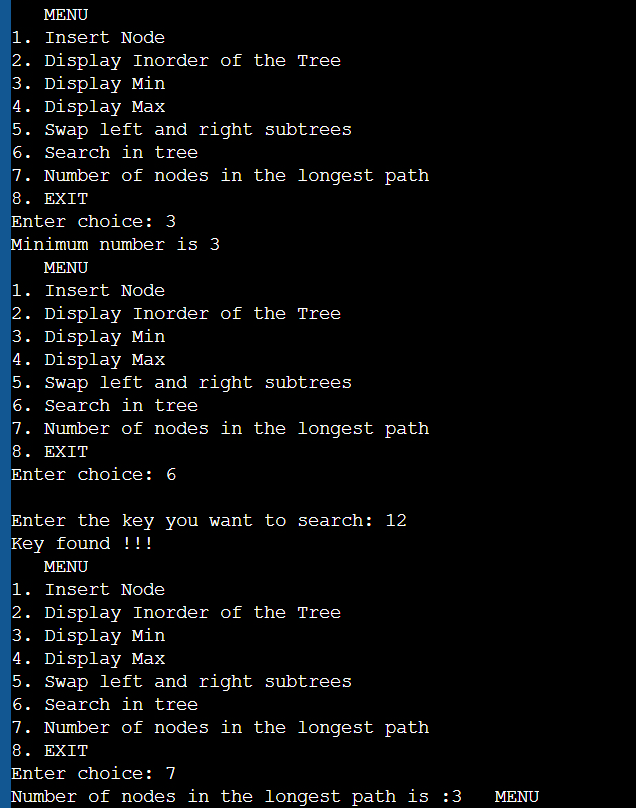
return 0;

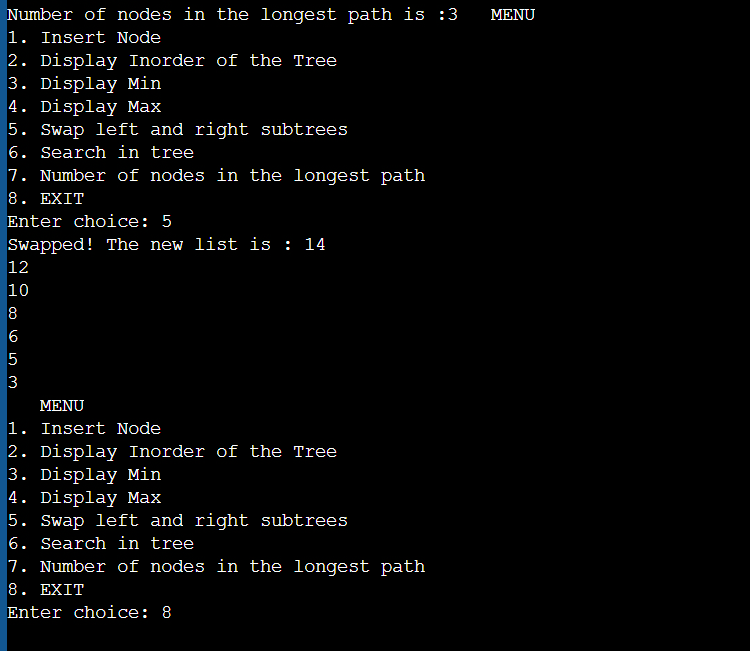
} **OUTPUT**

****

****

****

****

****