1) preorder,inorder,postorder

//C program for different tree transversals

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node\* left;

struct node\* right;

};

struct node\* newNode(int data)

{

struct node\* node = (struct node\*)

malloc(sizeof(struct node));

node->data = data;

node->left = NULL;

node->right = NULL;

return(node);

}

void printPostorder(struct node\* node)

{

if (node == NULL)

return;

// first recur on left subtree

printPostorder(node->left);

// then recur on right subtree

printPostorder(node->right);

// now deal with the node

printf("%d ", node->data);

}

void printInorder(struct node\* node)

{

if (node == NULL)

return;

/\* first recur on left child \*/

printInorder(node->left);

/\* then print the data of node \*/

printf("%d ", node->data);

/\* now recur on right child \*/

printInorder(node->right);

}

void printPreorder(struct node\* node)

{

if (node == NULL)

return;

/\* first print data of node \*/

printf("%d ", node->data);

/\* then recur on left sutree \*/

printPreorder(node->left);

/\* now recur on right subtree \*/

printPreorder(node->right);

}

int main()

{

struct node \*root = newNode(5);

root->left = newNode(6);

root->right = newNode(7);

root->left->left = newNode(8);

root->left->right = newNode(9);

printf("\nPreorder traversal of binary tree is \n");

printPreorder(root);

printf("\nInorder traversal of binary tree is \n");

printInorder(root);

printf("\nPostorder traversal of binary tree is \n");

printPostorder(root);

getchar();

return 0;

}

Output:

Preorder traversal of binary tree is 5 6 8 9 7

Inorder traversal of binary tree is 8 6 9 5 7

Postorder traversal of binary tree is 8 9 6 7 5

2)Insert or create binary search tree

// C program to demonstrate insert operation in binary search tree.

#include<stdio.h>

#include<stdlib.h>

struct node

{

int key;

struct node \*left, \*right;

};

struct node \*newNode(int item)

{

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

temp->key = item;

temp->left = temp->right = NULL;

return temp;

}

void inorder(struct node \*root)

{

if (root != NULL)

{

inorder(root->left);

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

inorder(root->right);

}

}

struct node\* insert(struct node\* node, int key)

{

/\* If the tree is empty, return a new node \*/

if (node == NULL) return newNode(key);

/\* Otherwise, recur down the tree \*/

if (key < node->key)

node->left = insert(node->left, key);

else if (key > node->key)

node->right = insert(node->right, key);

/\* return the (unchanged) node pointer \*/

return node;

}

int main()

{

/\* Let us create following BST

50

/ \

30 70

/ \ / \

20 40 60 80 \*/

struct node \*root = NULL;

root = insert(root, 50);

insert(root, 30);

insert(root, 20);

insert(root, 40);

insert(root, 70);

insert(root, 60);

insert(root, 80);

// print inoder traversal of the BST

inorder(root);

return 0;

}

Output:

20

30

40

50

60

70

80

3)Linear search

#include <stdio.h>

int search(int arr[], int n, int x)

{

int i;

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

if (arr[i] == x)

return i;

return -1;

}

int main(void)

{

int arr[] = { 2, 13, 46, 10, 40 };

int x = 10;

int n = sizeof(arr) / sizeof(arr[0]);

int result = search(arr, n, x);

(result == -1) ? printf("Element is not present in array")

: printf("Element is present at index %d",

result);

return 0;

}

Output:

Element is present at index 3

4)Binary search

#include <stdio.h>

int main()

{

int c, first, last, middle, n, search, array[100];

printf("Enter number of elements\n");

scanf("%d", &n);

printf("Enter %d integers\n", n);

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

scanf("%d", &array[c]);

printf("Enter value to find\n");

scanf("%d", &search);

first = 0;

last = n - 1;

middle = (first+last)/2;

while (first <= last) {

if (array[middle] < search)

first = middle + 1;

else if (array[middle] == search) {

printf("%d found at location %d.\n", search, middle+1);

break;

}

else

last = middle - 1;

middle = (first + last)/2;

}

if (first > last)

printf("Not found! %d isn't present in the list.\n", search);

return 0;

}

Output:

Enter the number of elements

6

Enter 6 integers

2

45

78

90

98

78

Enter the value to find

98

98 found at location 5