// C program for preorder inorder

#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;

printPostorder(node->left);

printPostorder(node->right);

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

}

void printInorder(struct node\* node)

{

if (node == NULL)

return;

printInorder(node->left);

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

printInorder(node->right);

}

void printPreorder(struct node\* node)

{

if (node == NULL)

return;

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

printPreorder(node->left);

printPreorder(node->right);

}

int main()

{

struct node \*root = newNode(1);

root->left = newNode(2);

root->right = newNode(3);

root->left->left = newNode(4);

root->left->right = newNode(5);

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

printPreorder(root);

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

printInorder(root);

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

printPostorder(root);

getchar();

return 0;

}

//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()

{

struct node \*root = NULL;

root = insert(root, 5);

insert(root, 3);

insert(root, 2);

insert(root, 4);

insert(root, 7);

insert(root, 6);

insert(root, 8);

// print inoder traversal of the BST

inorder(root);

return 0;

}

Output:

2

3

4

5

6

7

8

// C program for linear search algorithm//

#include<stdio.h>

#include<conio.h>

void main()

{

int a[10],i,size,item,pos,flag=0;

printf("\n Enter the size of an array: ");

scanf("%d",&size);

printf("\n Enter the elements of the array: ");

//LOOP TO STORE THE ELEMENTS

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

{

scanf("%d",&a[i]);

}

printf("\n Enter the element to be searched: ");

scanf("%d",&item);

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

{

if(item==a[i])

{

pos=i;

flag=1;

break;

}

}

if(flag==1)

printf("\n The element is in the list and its position is: %d",pos+1);

else

printf("\n The element is not found");

}

// C program for binary search algorithm//

#include<stdio.h>

int main()

{

int arr[50],i,n,x,flag=0,first,last,mid;

printf("Enter size of array:");

scanf("%d",&n);

printf("\nEnter array element(ascending order)\n");

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

scanf("%d",&arr[i]);

printf("\nEnter the element to search:");

scanf("%d",&x);

first=0;

last=n-1;

while(first<=last)

{

mid=(first+last)/2;

if(x==arr[mid]){

flag=1;

break;

}

else

if(x>arr[mid])

first=mid+1;

else

last=mid-1;

}

if(flag==1)

printf("\nElement found at position %d",mid+1);

else

printf("\nElement not found");

return 0;

}