1.Write a C program to print preorder, inorder, and postorder traversal on Binary Tree.

#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;

}

Output1

Preorder of binary tree is

1 2 4 5 3

In order of binary tree is

4 2 5 1 3

Post order of binary tree is

4 5 2 3 1

2.Write a C program to create (or insert) and inorder traversal on 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 (node == NULL) return newNode(key);

if (key < node->key)

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

else if (key > node->key)

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

return node;

}

int main()

{

struct node \*root = NULL;

root = insert(root, 2);

insert(root, 5);

insert(root, 1);

insert(root, 6);

insert(root, 7);

insert(root, 3);

insert(root, 4);

inorder(root);

return 0;

}

Output:-

1

2

7

3

4

5

6

3.Write a C program for linear search algorithm.

#include <stdio.h>

long linear\_search(long [], long, long);

int main()

{

long array[100], search, c, n, position;

printf("enter number of elements in array\n");

scanf("%ld", &n);

printf("Input %d numbers\n", n);

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

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

printf("enter a number to search\n");

scanf("%ld", &search);

position = linear\_search(array, n, search);

if (position == -1)

printf("%d isn't present in the array.\n", search);

else

printf("%d is present at location %d.\n", search, position+1);

return 0;

}

long linear\_search(long a[], long n, long find) {

long c;

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

if (a[c] == find)

return c;

Output:-

Enter number of elements in array

5

Enter 5 integer(s)

5

6

7

8

9

Enter a number to search

8

8 is present at location 3.

4.Write a C program for binary search algorithm

#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 number of elements

5

Enter 5 integers

5

6

7

8

9

8 found at location 3.