**DSA PRACTICAL FILE**

(CodeChef & online GDB as C language IDE is used for all the programs)

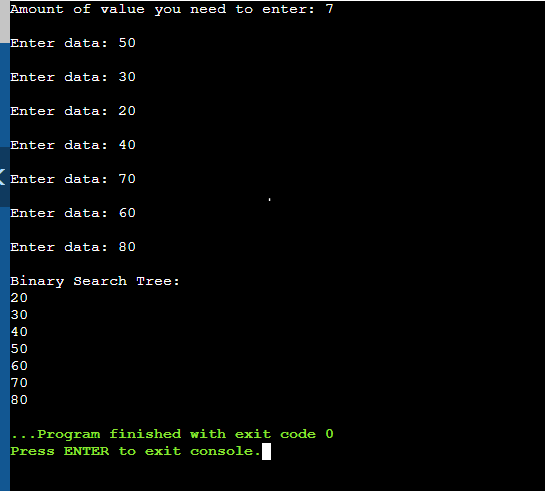
Practical File programs from 21 & 22

Name: Hardik Chhabra

ECAM-1

Roll no.: 2020UEA6504

**OUTPUT-21**



**Practical-21**

**Q. Write a program to construct a 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("\n%d", 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;

int n;

printf("Amount of value you need to enter: ");

scanf("%d",&n);

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

int item;

printf("\nEnter data: ");

scanf("%d",&item);

root=insert(root,item);

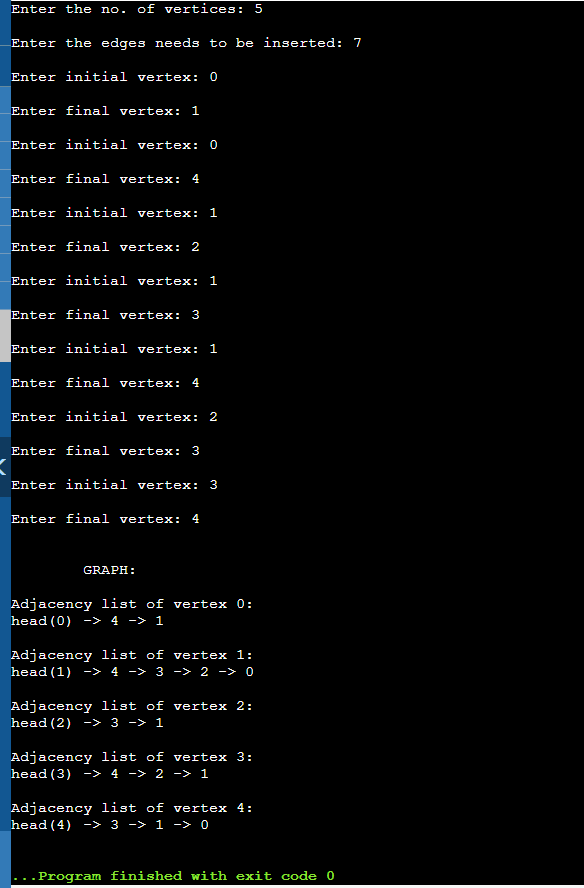
}

printf("\nBinary Search Tree: ");

inorder(root);

return 0;

}

**OUTPUT-22**

**Practical-22**

**Q. Write a program to construct a graph.**

#include <stdio.h>

#include <stdlib.h>

struct AdjListNode

{

int dest;

struct AdjListNode\* next;

};

struct AdjList

{

struct AdjListNode \*head;

};

struct Graph

{

int V;

struct AdjList\* array;

};

struct AdjListNode\* newAdjListNode(int dest)

{

struct AdjListNode\* newNode =

(struct AdjListNode\*) malloc(sizeof(struct AdjListNode));

newNode->dest = dest;

newNode->next = NULL;

return newNode;

}

struct Graph\* createGraph(int V)

{

struct Graph\* graph =

(struct Graph\*) malloc(sizeof(struct Graph));

graph->V = V;

graph->array =

(struct AdjList\*) malloc(V \* sizeof(struct AdjList));

int i;

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

graph->array[i].head = NULL;

return graph;

}

void addEdge(struct Graph\* graph, int src, int dest)

{

struct AdjListNode\* newNode = newAdjListNode(dest);

newNode->next = graph->array[src].head;

graph->array[src].head = newNode;

newNode = newAdjListNode(src);

newNode->next = graph->array[dest].head;

graph->array[dest].head = newNode;

}

void printGraph(struct Graph\* graph)

{

int v;

for (v = 0; v < graph->V; ++v)

{

struct AdjListNode\* pCrawl = graph->array[v].head;

printf("\nAdjacency list of vertex %d:\n", v);

printf("head(%d) ", v);

while (pCrawl)

{

printf("-> %d ", pCrawl->dest);

pCrawl = pCrawl->next;

}

printf("\n");

}

}

int main()

{

int V,n;

printf("Enter the no. of vertices: ");

scanf("%d",&V);

struct Graph\* graph = createGraph(V);

printf("\nEnter the edges needs to be inserted: ");

scanf("%d",&n);

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

int init,fin;

printf("\nEnter initial vertex: ");

scanf("%d",&init);

printf("\nEnter final vertex: ");

scanf("%d",&fin);

addEdge(graph, init, fin);

}

printf("\n\n\tGRAPH:\n");

printGraph(graph);

return 0;

}