DS Experiment – 9

Harshit Saraswat | 500097101 | B42

**A.1**

#include <stdio.h>

#include <stdlib.h>

//left child = 2i+1

//right child = 2i+2

//parent = ((i-1)/2)

int array\_empty = 1;

int \* array = NULL;

int size = 0;

int current\_index = 0;

int main()

{

printf("Enter Number of Nodes in Tree : ");

scanf("%d",&size);

array = (int \*)malloc(size\*sizeof(int));

printf("Enter 1st Node : ");

scanf("%d",&array[0]);

int input\_count = 1;

while(1){

int left\_child = 2\*current\_index + 1;

int right\_child= 2\*current\_index + 2;

printf("Enter Left Child of Index %d : ",current\_index);

scanf("%d",&array[left\_child]);

input\_count++;

if (input\_count == size)

break;

printf("Enter Right Child of Index %d : ",current\_index);

scanf("%d",&array[right\_child]);

input\_count++;

if (input\_count == size)

break;

current\_index++;

}

info\_of\_index();

printf("PreORDER Traversal of CBT is : ");

preorder(0);

return 0;

}

void info\_of\_index(){

int index = 0;

printf("Enter Index to find Information of : ");

scanf("%d",&index);

if (index == 0){

printf("Left Child of Root : %d\n",array[2\*index + 1]);

printf("Right Child of Root : %d\n",array[2\*index + 2]);

return;

}

printf("Parent of %d : %d\n",array[index],array[(index - 1)/2]);

printf("Left Child of %d : %d\n",array[index],array[2\*index + 1]);

printf("Right Child of %d : %d\n",array[index],array[2\*index + 2]);

}

void leaf\_nodes(){

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

if (2\*i + 1 > size){

printf("\nIndex %d is a Leaf node\n",i);

}

}

}

void preorder(int index){

printf("%d",array[index]);

if (2\*index + 1 < size){

preorder(2\*index + 1);

}

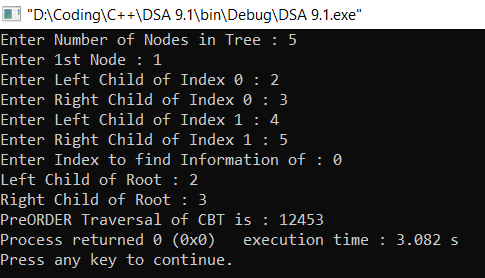
if (2\*index + 2 < size){

printf("%d",array[2\*index + 2]);

}

}

**OUTPUT :**



**A.2**

#include <stdio.h>

#include <stdlib.h>

//CREATE BINARY SEARCH TREE

struct node {

int data;

struct node \* r\_child;

struct node \* l\_child;

};

struct node \* root;

struct node \* ptr;

struct node \* new\_node;

int main()

{

int choice;

while (1){

printf("1 - Enter New Node\n2 - Print PreOrder Traversal\n3 - Print InOrder Traversal\n4 - Print PostOrder Traversal\n5 - Quit\nChoice : ");

scanf("%d",&choice);

switch(choice){

case 1 : insert\_new\_node();break;

case 2 : preorder\_traversal(root);break;

case 3 : inorder\_traversal(root);break;

case 4 : postorder\_traversal(root);break;

case 5 : exit(0);

}

printf("\n");

}

return 0;

}

void insert\_new\_node(){

new\_node = (struct node \*)malloc(sizeof(struct node));

printf("Enter A Value For the New Node to be Inserted : ");

scanf("%d",&new\_node->data);

new\_node->l\_child = NULL;

new\_node->r\_child = NULL;

if (root == NULL){

root = new\_node;

return;

}

ptr = root;

while (1){

if (new\_node->data < ptr->data || new\_node->data == ptr->data){

if (ptr->l\_child == NULL){

ptr->l\_child = new\_node;

break;

}

ptr = ptr->l\_child;

continue;

}

if (new\_node->data > ptr->data){

if (ptr->r\_child == NULL){

ptr->r\_child = new\_node;

break;

}

ptr = ptr->r\_child;

continue;

}

}

}

void print\_tree(struct node \* root){

if (root == NULL){

return;

}

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

if (root->l\_child!=NULL)

printf(" %d",root->l\_child->data);

if (root->r\_child!=NULL)

printf(" %d",root->r\_child->data);

printf("\n");

if (root->l\_child!=NULL)

print\_tree(root->l\_child);

if (root->r\_child!=NULL)

print\_tree(root->r\_child);

}

void inorder\_traversal(struct node \* parent){ //PERFECTIONE

if (parent->l\_child != NULL)

inorder\_traversal(parent->l\_child);

printf("| %d | ",parent->data);

if (parent->r\_child!=NULL){

inorder\_traversal(parent->r\_child);

}

}

void preorder\_traversal(struct node \* parent){

if (parent == NULL)

return;

printf("| %d | ",parent->data);

preorder\_traversal(parent->l\_child);

preorder\_traversal(parent->r\_child);

}

void postorder\_traversal(struct node \* parent){

if (parent == NULL)

return;

postorder\_traversal(parent->l\_child);

postorder\_traversal(parent->r\_child);

printf("| %d | ",parent->data);

}

**OUTPUT :**

