Name - Brijit Adak

Roll No - 21052410

Q1.WAP to implement Binary Tree using array and display all the nodes using another function.

#include<stdio.h>

void buildtree(int t[],int index,int value)

{

int ch,data;

t[index]=value;

printf("Do yiu have left child of %d (0/1) ? ",value);

scanf("%d",&ch);

if(ch)

{

printf("Enter the left child ");

scanf("%d",&data);

buildtree(t,2\*index+1,data);

}

printf("Do you have right child of %d (0/1) ? ",value);

scanf("%d",&ch);

if(ch)

{

printf("Enter the right child ");

scanf("%d",&data);

buildtree(t,2\*index+2,data);

}

}

void display(int t[])

{

for(int i=0;i<20;i++)

{

if(t[i]==-1)

printf("\_ ");

else

printf("%d ",t[i]);

}

}

int main()

{

int t[20],index,value;

for(int i=0;i<20;i++)

t[i]=-1;

printf("Enter the root node ");

scanf("%d",&value);

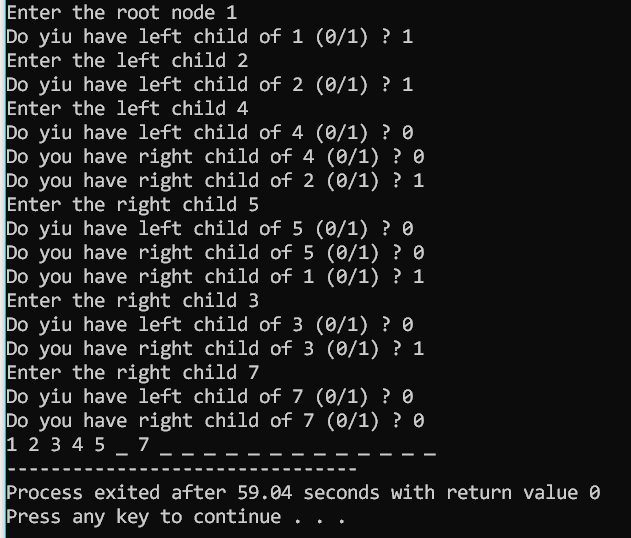
buildtree(t,0,value);

display(t);

return 0;

}

Output



Q2.WAP to implement Binary Tree using linked list and display all the nodes using another function.

#include<stdio.h>

#include<stdlib.h>

struct Tree

{

int data;

struct Tree \*lc,\*rc;

};

void buildtree(struct Tree \*ptr)

{

struct Tree \*New;

int ch;

printf("Enter value ");

scanf("%d",&ptr->data);

ptr->lc=NULL;

ptr->rc=NULL;

printf("Do you want to add left child of %d (0/1) ? ",ptr->data);

scanf("%d",&ch);

if(ch)

{

New=(struct Tree \*)malloc(sizeof(struct Tree));

ptr->lc=New;

buildtree(New);

}

printf("Do you want to add right child of %d (0/1) ? ",ptr->data);

scanf("%d",&ch);

if(ch)

{

New=(struct Tree \*)malloc(sizeof(struct Tree));

ptr->rc=New;

buildtree(New);

}

}

void display(struct Tree \*t)

{

if(t)

{

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

display(t->lc);

display(t->rc);

}

else

{

printf("\_ ");

}

}

int main()

{

struct Tree \*Root;

Root=(struct Tree \*)malloc(sizeof(struct Tree));

buildtree(Root);

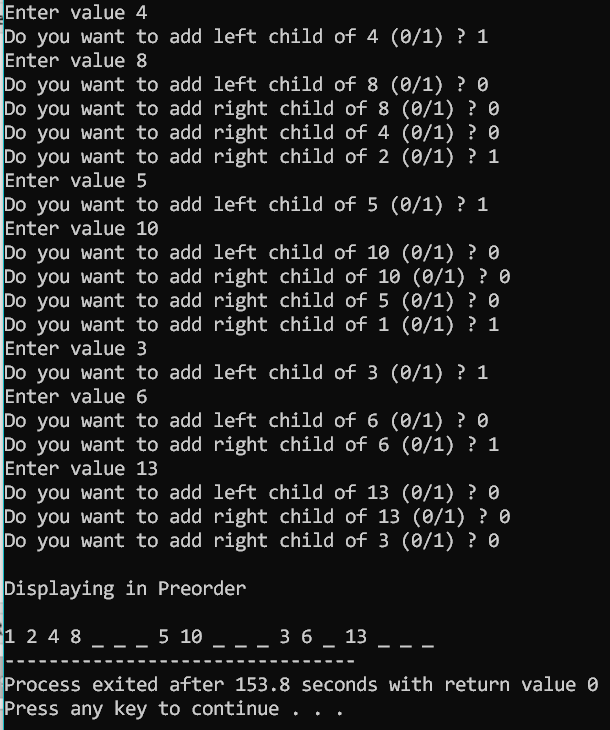
printf("\nDisplaying in Preorder\n\n");

display(Root);

return 0;

}

Output



Q3. WAP for inorder, preorder and postorder traversal using three different functions. Before that the binary tree is created using linked list.

#include<stdio.h>

#include<stdlib.h>

struct Tree

{

int data;

struct Tree \*lc,\*rc;

};

void buildtree(struct Tree \*ptr)

{

struct Tree \*New;

int ch;

printf("Enter value ");

scanf("%d",&ptr->data);

ptr->lc=NULL;

ptr->rc=NULL;

printf("Do you want to add left child of %d (0/1) ? ",ptr->data);

scanf("%d",&ch);

if(ch)

{

New=(struct Tree \*)malloc(sizeof(struct Tree));

ptr->lc=New;

buildtree(New);

}

printf("Do you want to add right child of %d (0/1) ? ",ptr->data);

scanf("%d",&ch);

if(ch)

{

New=(struct Tree \*)malloc(sizeof(struct Tree));

ptr->rc=New;

buildtree(New);

}

}

void display1(struct Tree \*t)

{

if(t)

{

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

display1(t->lc);

display1(t->rc);

}

else

{

printf("\_ ");

}

}

void display2(struct Tree \*t)

{

if(t)

{

display2(t->lc);

display2(t->rc);

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

}

else

{

printf("\_ ");

}

}

void display3(struct Tree \*t)

{

if(t)

{

display3(t->lc);

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

display3(t->rc);

}

else

{

printf("\_ ");

}

}

int main()

{

struct Tree \*Root;

Root=(struct Tree \*)malloc(sizeof(struct Tree));

buildtree(Root);

printf("\nDisplaying in Preorder\n\n");

display1(Root);

printf("\nDisplaying in Postorder\n\n");

display2(Root);

printf("\nDisplaying in Inorder\n\n");

display3(Root);

return 0;

}

Output

