**Name : Resham Landge**

**Roll No : 2339**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Assignment No : 3** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Title** : Create Binary tree and perform following operations:

a. Insert

b. Display

c. Depth of a tree

d. Display leaf-nodes

e. Create a copy of a tree

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include<iostream>

#include<math.h>

#include<string.h>

using namespace std;

int i;

class bintree

{

// Declares variable and functions as public

public :

typedef struct bin

{

char data[10];

struct bin \*left, \*right;

} node;

node \*qu[20], \*p, \*temp, \*newnode, \*root, \*temp1, \*p1, \*root1;

int j ,k, front, rear, h;

bintree() // Function to initialize root, root1, rear and front

{

root = root1 = NULL;

rear = front = -1;

}

int full(); // declares function full

int empty(); // declares function empty

void enqueue(node \*p); // declares function enqueue

node \*deqeue() // function to remove element from queue

{

if( !empty() )

{

temp = qu[front++];

p = temp;

return temp;

delete p;

}

cout<<" \n empty \n ";

}

void create(); // declares create function to create binary tree

void display(node \*root); // declares display function to display binary tree

void depth(node \*root); // declares depth function to show depth of BT

void disleaf(node \*temp); // declares disleaf function to show leaf node of

binary tree

void copy1()

{

root1 = copy(root);

}

node \*copy(node \*T) // Function to create copy of binary tree

{

node \*temp;

if(T == NULL)

return NULL;

else

{

temp = new node;

strcpy(temp->data, T->data);

temp->left = copy(T->left);

temp->right = copy(T->right);

return temp;

}

}

};

int bintree :: full() // function to check whether queue tree is full or not

{

if(rear == 19)

return 1;

return 0;

}

int bintree :: empty() // function to check whether queue is empty or not

{

if( ( (rear == -1) &&0 (front == -1) ) || (rear < front) )

return 1;

return 0;

}

void bintree :: enqueue(node \*p) // function to insert element in queue

{

if( !full() ) // determining whether queue is full or not

{

if((rear == -1) && (front == -1)) // checking whether the element to be

inserted is first element or not

{

rear = front = 0; // if element is first element then make rear

and front equal to zero

qu[rear] = p;

}

else

qu[++rear] = p;

}

else

cout<<" \n queue is full \n ";

}

void bintree :: create() // Function to create binary tree

{

newnode = new node; // initialize newnode

cout<<" \n Enter value ";

cin>>newnode->data;

newnode->left = NULL;

newnode->right = NULL;

if(root == NULL) // determine whether root is null or not

{

root = newnode; // if root is null, then root is equal to newnode

enqueue(newnode);

i = 1;

}

else

{

if(i == 1)

{

temp1 = deqeue();

temp1->left = newnode;

enqueue(newnode);

i++;

}

else

{

temp1->right = newnode;

enqueue(newnode);

i = 1;

}

}

}

void bintree :: display(node \*temp) // Function to display nodes of Binary Tree

{

cout<<" \n display \n ";

rear = front = -1;

p = temp;

cout<<" \n\t "<<p->data<<" \n";

enqueue(p); // Calling function enqueue

j=0;

do

{

h = 1;

while(h <= pow(2,j))

{

p1 = deqeue();

if(p1->left != NULL)

cout<<" \t "<<p1->left->data;

if(p1->right != NULL)

cout<<" \t "<<p1->right->data;

enqueue(p1->left);

enqueue(p1->right);

h++;

}

j++;

cout<<" \n ";

}while((p1->left != NULL) && (p1->right != NULL));

}

void bintree :: depth(node \*root) // Fuunction to display depth od binary tree

{

temp = root;

h=1;

while(temp->left != NULL) // Loop continues if temp->left is not NULL

{

temp = temp->left;

h++;

}

cout<<" \n Depth is "<<h;

}

void bintree :: disleaf(node \*temp) // Function to display leaf node of Binary Tree

{

if(temp != NULL) // Checks whether temp is null or not

{

if((temp->left == NULL) && (temp->right == NULL))

{

cout<<" \n "<<temp->data;

return;

}

disleaf(temp->left);

disleaf(temp->right);

}

}

/\* Main Function Contains Menu \*/

int main()

{

bintree b1;

char ans, ans1;

int c1, c2;

do

{

cout<<" \n 1- Create \n 2- display \n 3- display leaf node \n 4- depath of tree \n 5-

copy ";

cout<<"\n Enter ur choice : ";

cin>>c1;

switch(c1) // Switch Case to select your choice

{

case 1 : do

{

b1.create(); // Calling create function

cout<<" \n do u want to enter more node : ";

cin>>ans;

} while(ans == 'y'); // Loop continues if ans is equal to ‘y’

break;

case 2 : b1.display(b1.root); // Calling display function

break;

case 3 : b1.disleaf(b1.root); // Calling disleaf function

break;

case 4 : b1.depth(b1.root); // Calling depth function

break;

case 5 : b1.copy1(); // Calling copy1 function

cout<<" \n After copy created ";

b1.display(b1.root1); // Calling display function

break;

}

cout<<" \n Do u want to contiune (y/n) : ";

cin>>ans1;

} while(ans1 == 'y'); // Loop continues if ans is equal to ‘y’

return 0;

}

***Output-***

resham123@ubuntu:~/Desktop$ g++ ass3.cpp

resham123@ubuntu:~/Desktop$ ./a.out

1-create

2-display

3-display leaf node

4- depath of tree

5-copy

Enter ur choice 1

Enter value 1

do u want to enter more node y

Enter value 2

do u want to enter more node y

Enter value 3

do u want to enter more node y

Enter value 4

do u want to enter more node y

Enter value 5

do u want to enter more node n

Do u want to contiune (y/n) y

1-create

2-display

3-display leaf node

4- depath of tree

5-copy

Enter ur choice 2

display

1

2 3

4 5

Do u want to contiune (y/n) y

1-create

2-display

3-display leaf node

4- depath of tree

5-copy

Enter ur choice 3

4

5

Do u want to contiune (y/n) y

1-create

2-display

3-display leaf node

4- depath of tree

5-copy

Enter ur choice 4

Depth is 3

Do u want to contiune (y/n) y

1-create

2-display

3-display leaf node

4- depath of tree

5-copy

Enter ur choice 5

After copy created

display

1

2 3

4 5

Do u want to contiune (y/n) n