SIDDAGANGA INSTITUTE OF TECHNOLOGY, TUMAKURU-03

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

Data Structures and Applications (3RIS02)

Assignment

Academic Year: ODD SEM 2021-22

Submitted by,

Ritwik Raj	1SI20IS042
Harsh Tayal	1SI20IS014
Alok Kumar	1SI20IS001
Ishan Verma	1SI20IS015

FACULTY COORDINATOR

Mrs. Sharmila S P

TABLE OF CONTENTS

Chapter- 1

Problem Statement

- **Family Tree**

- **Problem Statement –** Implement a program to construct a tree representing hierarchy of a family. Choice of user can be a male dominant or female dominant. Read the names, construct the tree and display results in level order only.

- **Description –** A tree is to be constructed using names of the family members of a family in a hierarchal manner. The tree can start either from a boy child or a girl child and will go in backward direction towards their ancestors (Example -Father, Grandfather, Great Grandfather etc.). Once the tree is constructed a level order traversal of the tree is to be performed to print each generation separately.

- **Problem Understanding –** A binary tree is to be constructed using the names of the family members. Once the tree is constructed the tree is to be traversed in level order traversal i.e., one generation should be printed altogether from left to right in a tree.

Chapter- 2

Methodology

Tree Construction:

A binary tree is to be constructed using strings which will act as the names of the family members. To construct the tree, we have taken inputs from the user in the form of names of the members they want to insert following up with their gender and then the side of the family to which they belong i.e. paternal side or the maternal side of the previous person. For construction of the tree, we have used the given gender and the family side input by the user to determine to which side of the node i.e. the previous person, the next string (name input) is to be attached. According to our program, if the person is of male gender(M) and belongs to the paternal side(P) of the previous person then he is attached to the left end of the node and if the gender input is female(F) and maternal(M), then she is attached to the right side of the previous node.

Level order traversal:

- **Queue Approach:** In this approach for the level order traversal a queue is used. First the root is inserted into the queue followed by a null element into the queue. This null element acts as a delimiter. Next, popping from the top of the queue is done and add its left and right nodes are pushed to the to the end of the queue and then print at the top of the queue. This process is continued till the queue becomes empty.
- **Recursive Approach:** In this approach for the level order traversal, we use recursive function call to get the desired result. For this approach to level order traversal, we also need the height of the tree which is then again calculated using the recursive approach.

Data Structures Used:

- Tree data structure – This data structure is used for the construction of the family tree.
- Queues – Queues is used here for the level order traversal of the family tree constructed.
- Linked Lists – It is used for the construction of the queue which is further used to traverse the tree in level order fashion.
- Arrays – Arrays are used in the program in the form of character arrays to store the names of the family members given as input by the user for construction of the tree.

Chapter- 3

Solution to the Problem

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
//structure declaration for tree
struct node
{
    char data[25];
    struct node *left, *right;
};
typedef struct node *NODE;
//structure declaration for queue
struct nodeq
{
    struct node *item;
    struct nodeq *next;
};
typedef struct nodeq *NODEQ;
//insrertion at the end function for the queue
NODEQ insertlast(NODEQ first, NODE ele)
{
    NODEQ newnode, temp;
    //dynamic allocation of memory for the new node of queue
    newnode = (NODEQ)malloc(sizeof(struct nodeq));
    newnode->item = ele;
    newnode->next = NULL;
    if (first == NULL)
    {
        first = newnode;
        return first;
```

```
    }
    temp = first;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newnode;
    return first;
}
//function to delete the first element from the queue
NODEQ deletefirst(NODEQ first)
{
    NODEQ temp;
    if (first == NULL)
        return first;
    temp = first;
    first = first->next;
    free(temp);
    return first;
}
//level order traversal of the tree using queues
void levelorder(NODE root)
{
    if (root == NULL)
        return;
    NODEQ queue;
    queue = NULL;
    //first the root is inserted into the queue
    queue = insertlast(queue, root);
    //insertion of NULL which acts as a delimiter
    queue = insertlast(queue, NULL);
    while (queue != NULL)
    {
        NODEQ front;
```

```
        front = queue;
        if (queue->item != NULL)
        {
            printf("%s ", queue->item->data);
            //pushing of the left children of the popped node
            if (front->item->left != NULL)
                queue = insertlast(queue, front->item->left);
            //pushing of the right children of the popped node
            if (front->item->right != NULL)
                queue = insertlast(queue, front->item->right);
        }
        //pushing the elements left over
        else if (queue->next != NULL)
        {
            queue = insertlast(queue, NULL);
            printf("\n");
        }
        queue = deletefirst(queue);
    }
}
//function to insert a node in the tree
NODE insert(NODE root, char gender, char name[])
{
    NODE temp, newnode, temp1;
    char ele;
    temp = root;
    //dynamic memory allocation of the node which is to be
inserted
    newnode = (NODE)malloc(sizeof(struct node));
    strcpy(newnode->data, name);
    newnode->left = newnode->right = NULL;
    if (root == NULL)
    {
```

```
        root = newnode;
        return root;
    }
    while (temp != NULL)
    {
        temp1 = temp;
        //relationship with the previous node
        //determines which side node will be attached
        printf("Relation with %s (P for Paternal / M for Maternal)
:", temp->data);
        getchar();
        scanf("%c", &ele);
        if (ele == 'p' || ele == 'P')
            temp = temp->left;
        else if (ele == 'm' || ele == 'M')
            temp = temp->right;
    }
    //gender determination
    //another factor to determine the side to be attached
    if (gender == 'm' || gender == 'M')
        temp1->left = newnode;
    else
        temp1->right = newnode;
    return root;
}
//function to calculate the height of the tree
int height(NODE root)
{
    if (root == NULL)
        return 0;
    //recursive call for the height of the left side of the tree
    int lheight = height(root->left);
    //recursive call for the height of the right side of the tree
```

```
    int rheight = height(root->right);
    //determination of the height by
    //comparing the left and right height of the tree
    if (lheight > rheight)
        return lheight + 1;
    else
        return rheight + 1;
}
//function to print the current level of the tree
void printCurrentLevel(NODE root, int level)
{
    if (root == NULL)
    {
        return;
    }
    //prints the elements in the current level
    if (level == 1)
    {
        printf("%s ", root->data);
    }
    //recursive call to print next level left side
    printCurrentLevel(root->left, level - 1);
    //recursive call to print next level right side
    printCurrentLevel(root->right, level - 1);
}
//function which calls the printCurrentLevel
//it gives the proper arguments to print the current level
void LevelOrderRecursion(NODE root)
{
    int h = height(root);
    for (int i = 1; i <= h; i++)
    {
```

```
        printCurrentLevel(root, i);
        printf("\n");
    }
}
//beginning of the main function
int main()
{
    int choice;
    char name[25], gender;
    //initialisation of the root of the tree
    NODE root = NULL;
    while (1)
    {
        printf(" 1.Insert\n 2.Level Order using queues\n 3.Level
order using recursion\n 4. Exit\n Enter your choice :");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            //element insertion in tree
            printf("Enter the element to be inserted :");
            getchar();
            scanf("%s", name);
            printf("Enter your gender:(M for Male / F for Female)
:");
            getchar();
            scanf("%c", &gender);
            root = insert(root, gender, name);
            break;
        case 2:
            //level order using queues function call
            levelorder(root);
            printf("\n");
```

```
            break;
        case 3:
            //level order using recursion function call
            LevelOrderRecursion(root);
            printf("\n");
            break;
        case 4:
            //terminate the program
            exit(0);
            break;
        default:
            printf("Invalid Input....\n");
            break;
        }
    }
    return 0;
}
//end of the main function
```

Chapter- 4

Screenshots

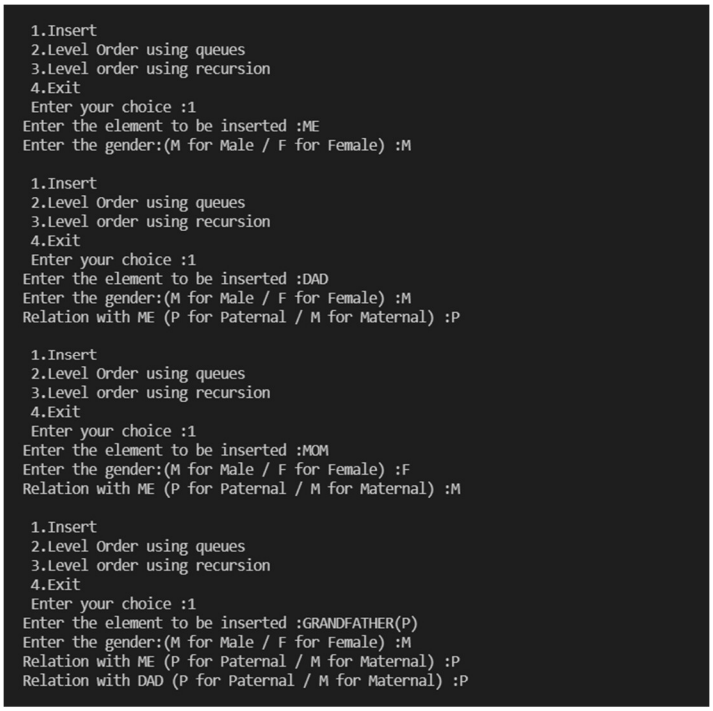

Fig. 1 – Insertion of the elements into the tree till level 2

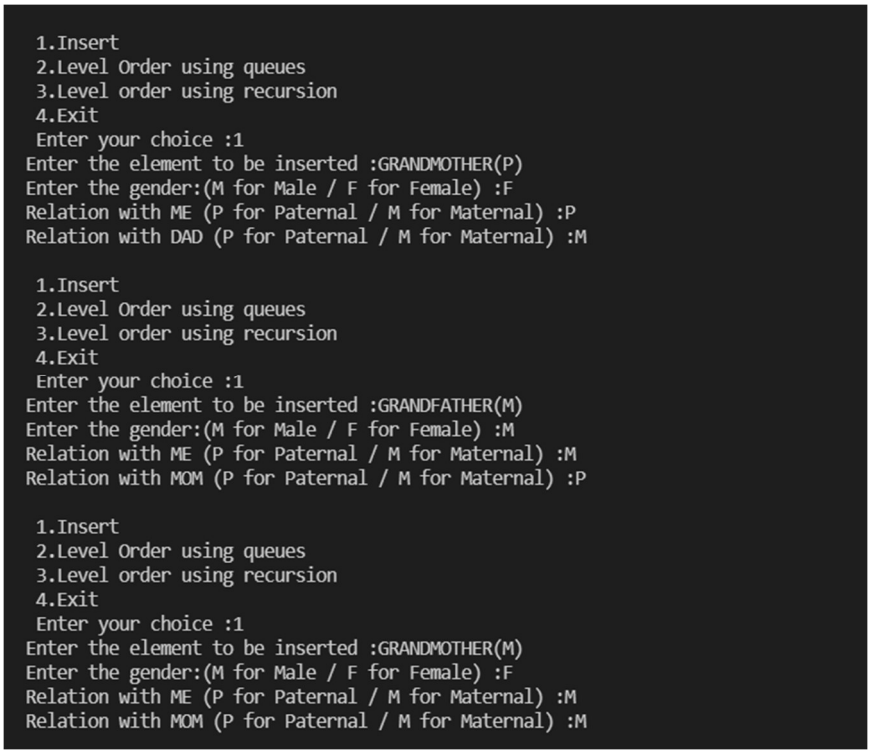

Fig. 2 – Insertion of the elements into the tree on level 2 continued

```
 1.Insert
 2.Level Order using queues
 3.Level order using recursion
 4.Exit
 Enter your choice :2
ME
DAD MOM
GRANDFATHER(P) GRANDMOTHER(P) GRANDFATHER(M) GRANDMOTHER(M)

 1.Insert
 2.Level Order using queues
 3.Level order using recursion
 4.Exit
 Enter your choice :3
ME
DAD MOM
GRANDFATHER(P) GRANDMOTHER(P) GRANDFATHER(M) GRANDMOTHER(M)


 1.Insert
 2.Level Order using queues
 3.Level order using recursion
 4.Exit
 Enter your choice :4
```

Fig. 3 – Level order traversal of the tree using queues as well as using recursion

```
 1.Insert
 2.Level Order using queues
 3.Level order using recursion
 4.Exit
 Enter your choice :1
Enter the element to be inserted :ME
Enter the gender:(M for Male / F for Female) :M

 1.Insert
 2.Level Order using queues
 3.Level order using recursion
 4.Exit
 Enter your choice :1
Enter the element to be inserted :DAD
Enter the gender:(M for Male / F for Female) :M
Relation with ME (P for Paternal / M for Maternal) :P

 1.Insert
 2.Level Order using queues
 3.Level order using recursion
 4.Exit
 Enter your choice :1
Enter the element to be inserted :MOM
Enter the gender:(M for Male / F for Female) :F
Relation with ME (P for Paternal / M for Maternal) :M
```

Fig. 4 – Insertion of elements of a nucleic family

```
 1.Insert
 2.Level Order using queues
 3.Level order using recursion
 4.Exit
 Enter your choice :2
ME
DAD MOM

 1.Insert
 2.Level Order using queues
 3.Level order using recursion
 4.Exit
 Enter your choice :3
ME
DAD MOM
```

Fig. 5 – Level order displays

Chapter- 5

Conclusion

A tree is a non-linear and hierarchical data structure consisting of a collection of nodes such that each node of the tree stores a value, a list of reference to nodes (the “children”).

Through this problem we came to learn about the various form of data structures used to store data, be it an integer data or any other. We also learned about the various ways of traversing through a tree (travelling through a tree) out of which the level order traversal being the utmost priority for us to showcase the different generations of a family.

Level order traversal is done in various ways, basically two ways,

- By using queues
- By using recursions

We also learned the differences between the two ways of level order traversal like the method using queues is faster way to traverse as compared to the method using recursions. The recursive approach also requires two support functions, one which calculate the height of the tree and one which traverses through the levels. This recursive approach also gave us the opportunity to learn more about the height of the binary tree.

Finally, this assignment taught us teamwork and how to convey our thought process to our team members. It was a joy ride throughout with its own ups and downs. We would like to thank our faculty coordinator also for giving us this opportunity to work as a team, showcase our skills and be motivated to learn something new every day.

References

[1] GeeksForGeeks

[2] Youtube.com

[3] Various reference textbooks