Practical-8

1. Level order traversal.

Aim:

To implement trees and traversing it in level order.

Theory:

Implemented tree using linked list also we used queue using linked list for level order traversal.

Code:

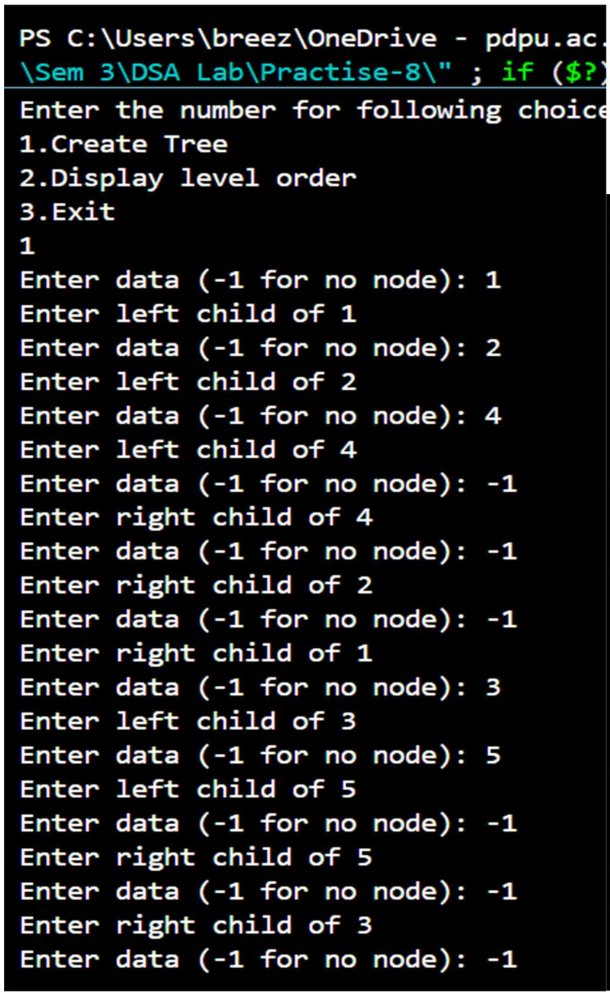
```
#include<stdio.h>
#include<stdlib.h>

struct node
{
  int data;
  struct node * right,* left;
};

struct queueNode
{
  struct node *treeNode;
  struct queueNode *next;
};

struct queueNode *forw=NULL,*rear=NULL;

void enqueue(struct node *treeNode)
{
  struct queueNode *newQ=(struct queueNode *)malloc(sizeof(struct queueNode));
  newQ->treeNode=treeNode;
```

```
 newQ->next=NULL;
 if(rear==NULL)
 {
  forw=rear=newQ;
 }
 else
 {
  rear->next=newQ;
  rear=newQ;
 }
}

struct node* dequeue()
{
 if(forw==NULL)
  return NULL;
 struct queueNode *temp=forw;
 struct node *treeNode=temp->treeNode;
 forw=forw->next;
 if(forw==NULL)
  rear=NULL;
 return treeNode;
}

int isEmpty()
{
 return (forw==NULL);
}
```

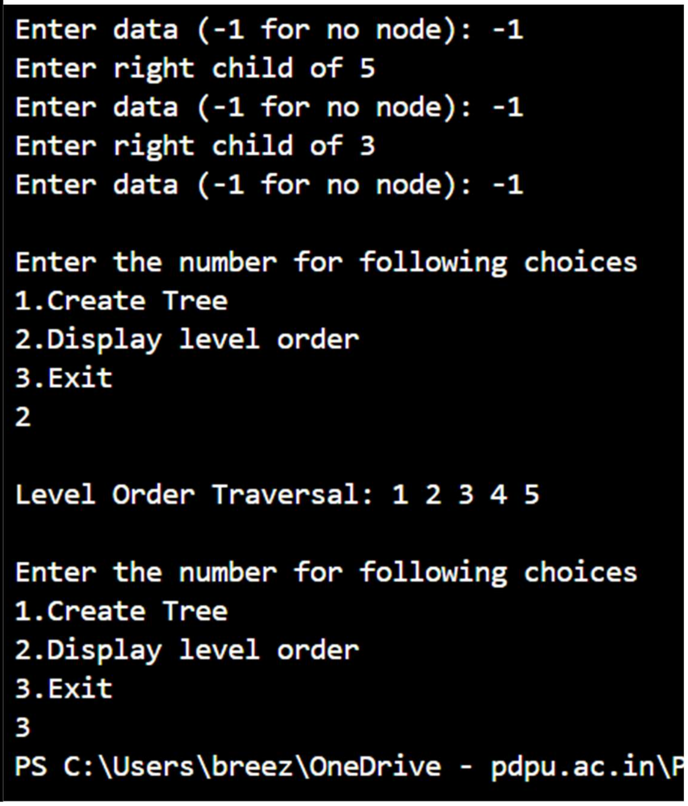
```
struct node* createNode(int data)
{
 struct node* newNode=(struct node *)malloc(sizeof(struct node));
 newNode->data=data;
 newNode->left=newNode->right=NULL;
 return newNode;
}

struct node* createTree()
{
 int data;
 printf("Enter data (-1 for no node): ");
 scanf("%d",&data);
 if(data==-1) return NULL;
 struct node* root=createNode(data);
 printf("Enter left child of %d\n",data);
 root->left=createTree();
 printf("Enter right child of %d\n",data);
 root->right=createTree();
 return root;
}

void displayLevelOrder(struct node *root)
{
 if(root==NULL)
 {
  printf("Tree is empty!\n");
  return;
 }
 enqueue(root);
 printf("\nLevel Order Traversal: ");
 while(!isEmpty())
 {
  struct node *current=dequeue();
  printf("%d ",current->data);
  if(current->left!=NULL)
   enqueue(current->left);
  if(current->right!=NULL)
   enqueue(current->right);
 }
 printf("\n");
}
int main()
{
 struct node *root=NULL;
 int a;
 while(1)
 {
  printf("\nEnter the number for following choices \n1.Create Tree \n2.Display level
order \n3.Exit\n");
  scanf("%d",&a);
  switch(a)
  {
   case 1:
    root=createTree();
    break;
   case 2:
    displayLevelOrder(root);
```

```
      break;
    case 3:
      exit(0);
      break;
    default:
      printf("Invalid Choice");
      break;
  }
 }
 return 0;
}
```

Output:



```
PS C:\Users\breez\OneDrive - pdpu.ac.
\Sem 3\DSA Lab\Practise-8\" ; if ($?)
Enter the number for following choice
1.Create Tree
2.Display level order
3.Exit
1
Enter data (-1 for no node): 1
Enter left child of 1
Enter data (-1 for no node): 2
Enter left child of 2
Enter data (-1 for no node): 4
Enter left child of 4
Enter data (-1 for no node): -1
Enter right child of 4
Enter data (-1 for no node): -1
Enter right child of 2
Enter data (-1 for no node): -1
Enter right child of 1
Enter data (-1 for no node): 3
Enter left child of 3
Enter data (-1 for no node): 5
Enter left child of 5
Enter data (-1 for no node): -1
Enter right child of 5
Enter data (-1 for no node): -1
Enter right child of 3
Enter data (-1 for no node): -1
```



```
Enter data (-1 for no node): -1
Enter right child of 5
Enter data (-1 for no node): -1
Enter right child of 3
Enter data (-1 for no node): -1

Enter the number for following choices
1.Create Tree
2.Display level order
3.Exit
2

Level Order Traversal: 1 2 3 4 5

Enter the number for following choices
1.Create Tree
2.Display level order
3.Exit
3
PS C:\Users\breez\OneDrive - pdpu.ac.in\P
```

Link for all codes:

https://github.com/PanavPatel06/DSA-Lab/tree/main/Practise-8