**Data Structures**

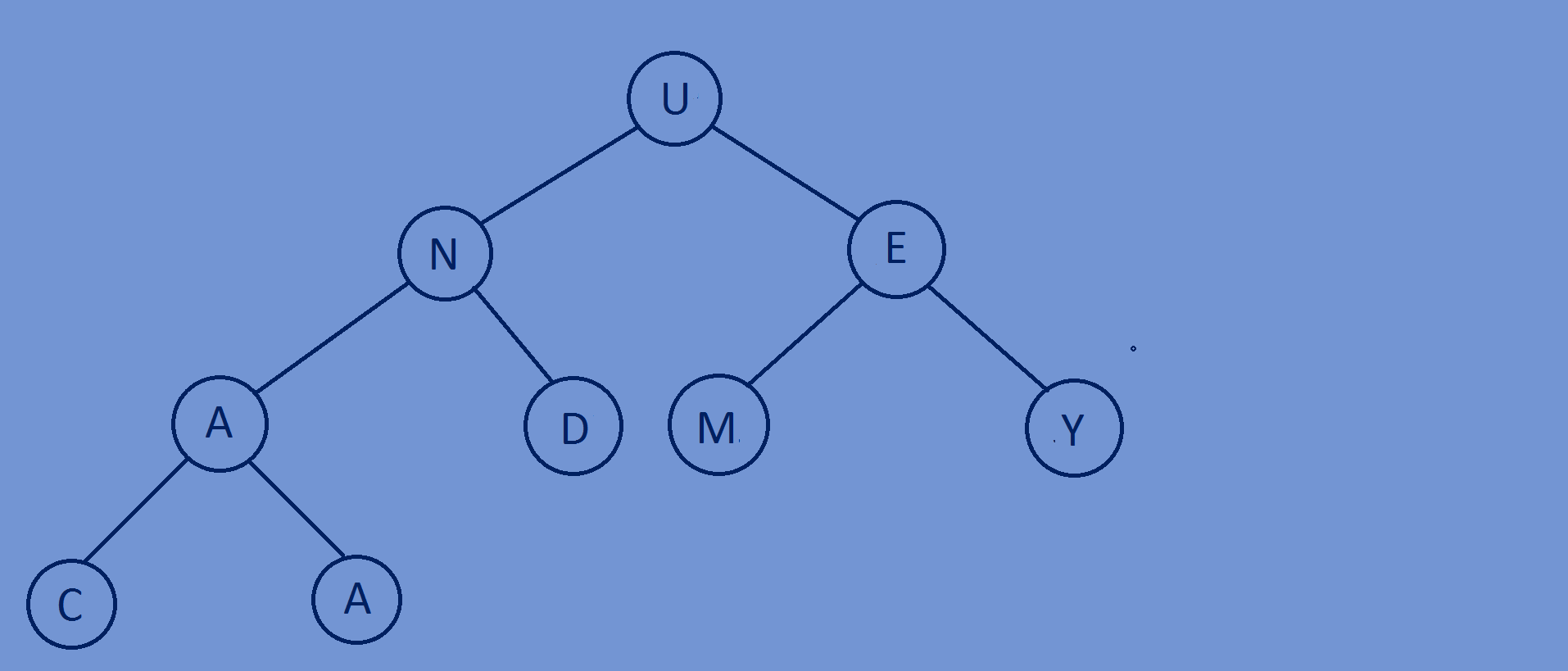
Implementation of Breadth first Traversal

**Breadth first traversal:**

* The process of visiting all the nodes of a tree level by level is called as breadth

First traversal.

* Breadth first traversal for the following tree is U N E A D M Y C A

****

**Implementation of Breadth first traversal:**

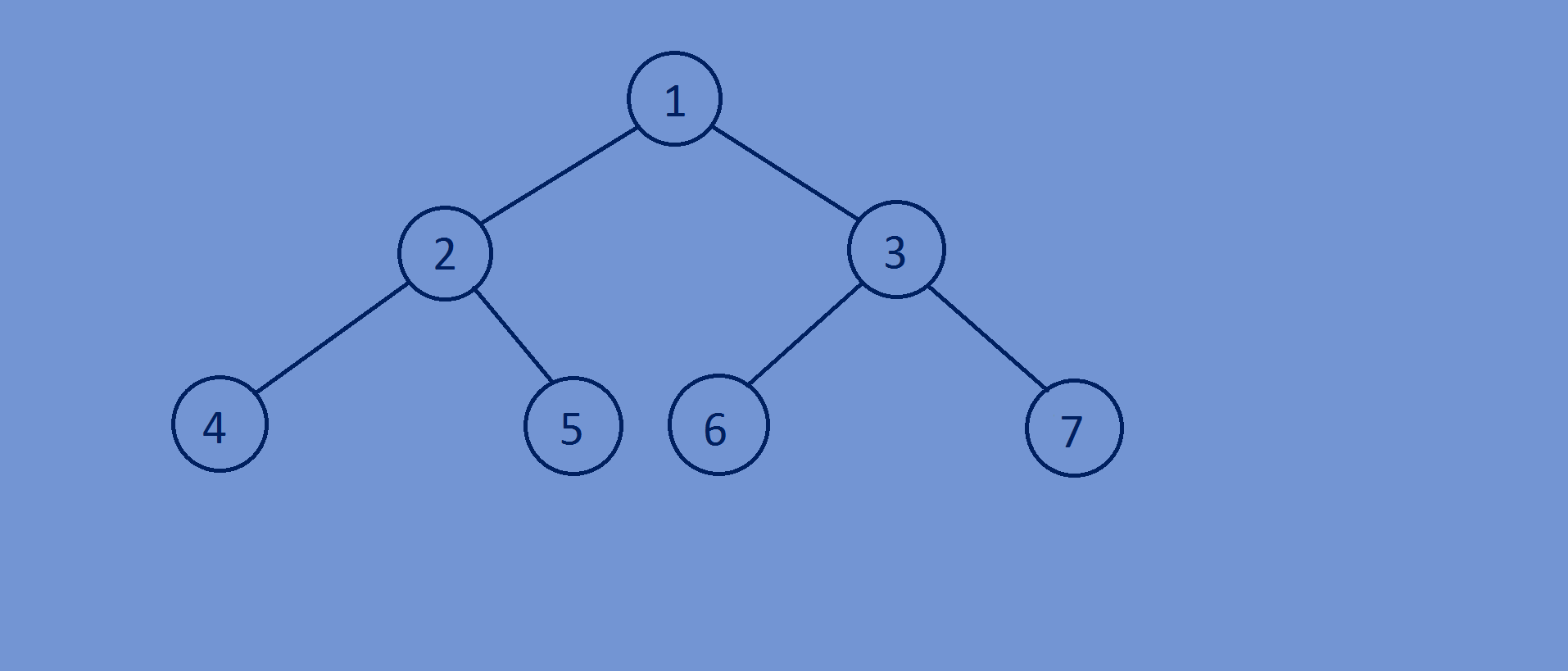
* First print the data of the root node, if left pointer of the root node is not

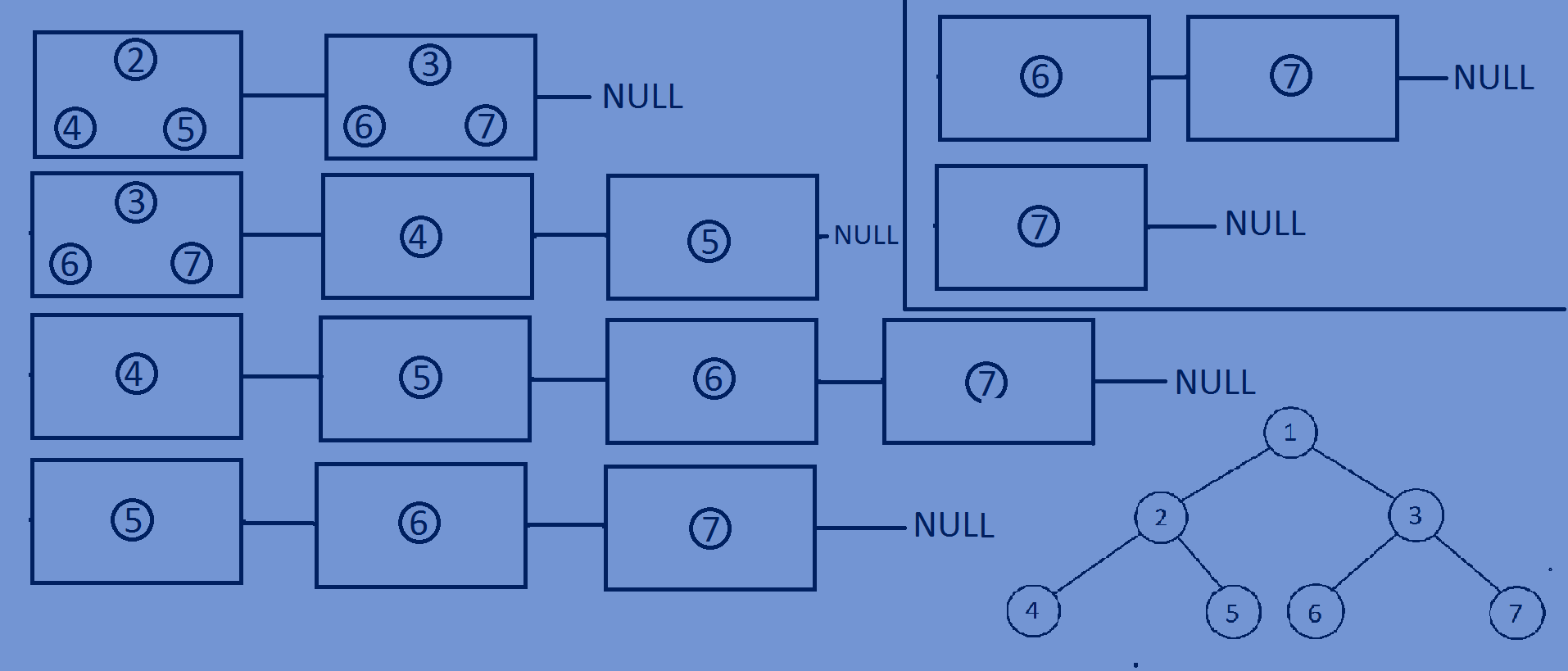
Equal to null then insert the left subtree of the root node into

Queue(enqueue) and if the right pointer of the root node is not equal to

Null then insert the right subtree of the root node into Queue(enqueue).

At the end of each iteration make the data(tree) of the first node as the

binary tree And then pop(dequeue) the first node of the queue.



**Function for printing of all the the elements of a tree level by level:**

**void** LevelOrder(Btree \* root)  
{  
 lin\_list \*q=NULL;  
 Btree \*var=root;  
 **while**(var!=NULL) {  
 printf("%d ",var->data1);  
 **if**(var->left!=NULL)  
 {  
 q=enqueue(q,var->left);  
 }  
 **if**(var->right!=NULL)  
 {  
 q=enqueue(q,var->right);  
 }  
 **if**(q==NULL){  
 **break**;  
 }  
 var=q->data;  
 q=dequeue(q);  
 }  
}

**Whole program:**

#include<stdio.h>  
#include<stdlib.h>  
  
//creating a Binary tree node.  
**typedef struct** Btree{  
 **int** data1;  
 **struct** Btree \*left;  
 **struct** Btree \*right;  
}Btree;  
//creating a linked list node.  
**typedef struct** lin\_list{  
 Btree \*data;  
 **struct** lin\_list \*next;  
}lin\_list;  
//function for inserting nodes at the end of a linked list  
lin\_list \*enqueue(lin\_list \*head,Btree \*tree){  
 lin\_list \*newnode=(lin\_list\*)malloc(**sizeof**(lin\_list));  
 newnode->data=tree;  
 newnode->next=NULL;  
 lin\_list \*temp=head;  
 **if**(head==NULL){  
 head=newnode;  
 }  
 **else** {  
 **while** (temp->next != NULL) {  
 temp = temp->next;  
 }  
 temp->next = newnode;  
 }  
 **return** head;  
}  
//function for popping of first node of a linked list  
lin\_list \*dequeue(lin\_list \*head){  
 lin\_list \*temp=head;  
 head=head->next;  
 free(temp);  
 **return** head;  
}  
//create newnode  
Btree \*newnode(**int** data){  
 Btree \*tree=(Btree\*)malloc(**sizeof**(Btree));  
 tree->data1=data;  
 tree->left=NULL;  
 tree->right=NULL;  
 **return** tree;  
}  
//level order traversal of a tree  
**void** LevelOrder(Btree \* root)  
{  
 lin\_list \*q=NULL;  
 Btree \*var=root;  
 **while**(var!=NULL) {  
 printf("%d ",var->data1);  
 **if**(var->left!=NULL)  
 {  
 q=enqueue(q,var->left);  
 }  
 **if**(var->right!=NULL)  
 {  
 q=enqueue(q,var->right);  
 }  
 **if**(q==NULL){  
 **break**;  
 }  
 var=q->data;  
 q=dequeue(q);  
 }  
}  
//main function  
**int** main(){  
 Btree \*tree;  
 tree=newnode(1);  
 tree->left=newnode(2);  
 tree->right=newnode(3);  
 tree->left->left=newnode(4);  
 tree->left->right=newnode(5);  
 tree->right->left=newnode(6);

tree->right->right=newnode(7);  
 LevelOrder(tree);  
 **return** 0;  
}

**Output:**

1 2 3 4 5 6 7