**Data Structures**

Implementation of Deapth first Traversal

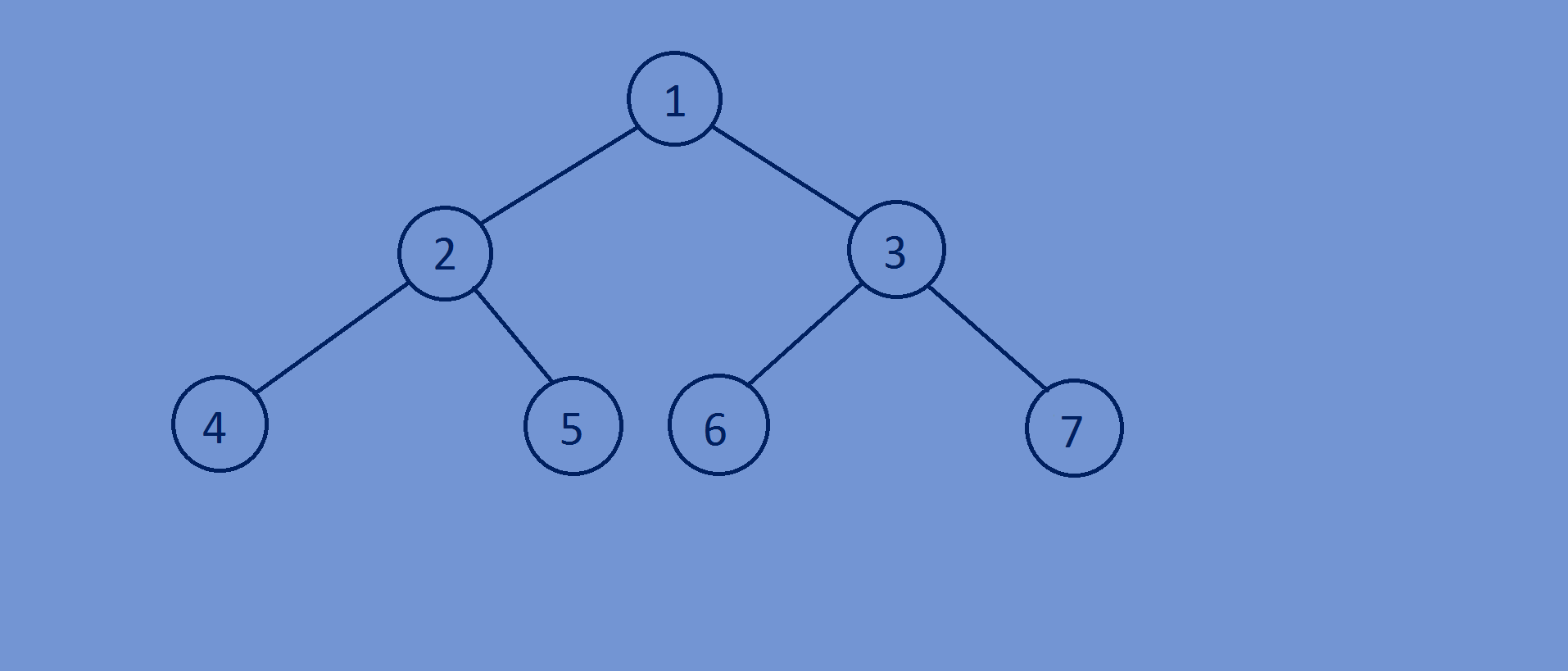
**Depth first traversal:**

* There are three ways of traversing in case of depth first traversal

1)Preorder traversal(root->left->right).

2)Inorder traversal(left->root->right).

3)Postorder traversal(left->right->root).

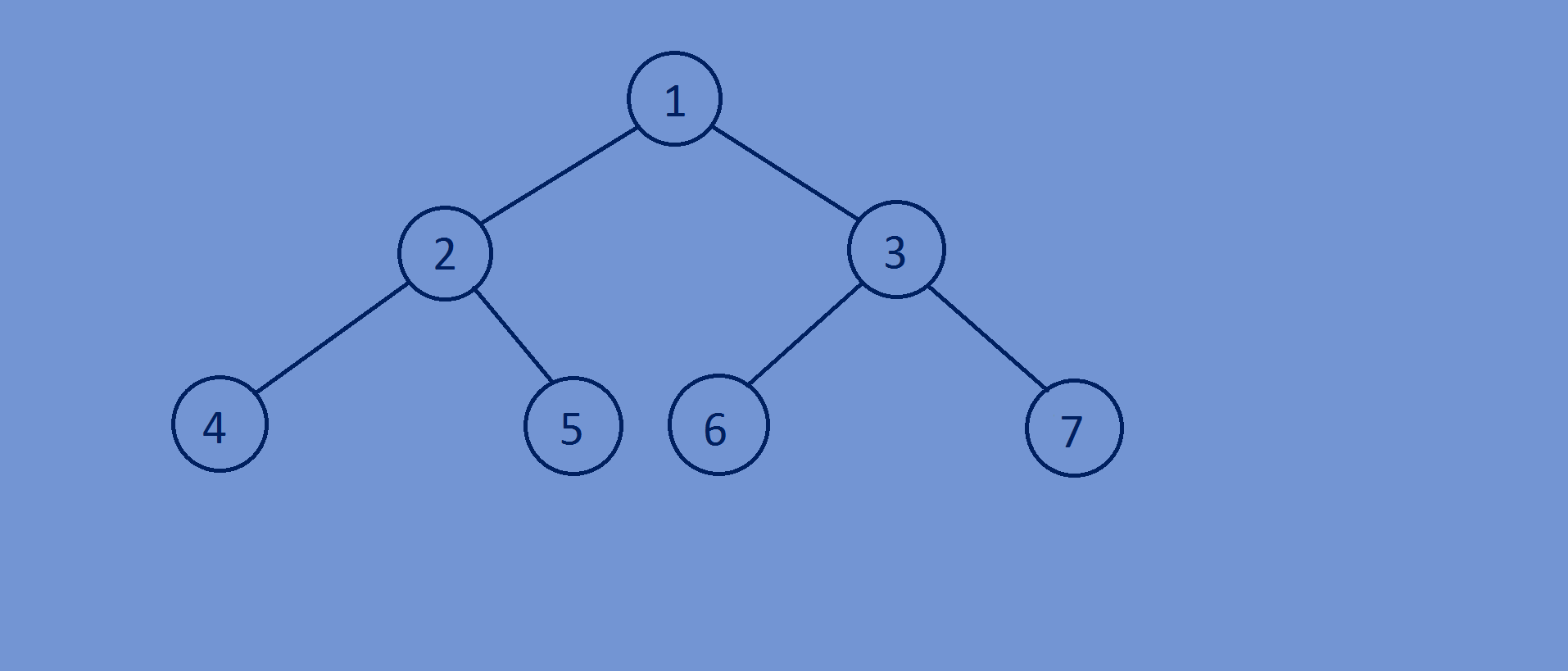


**Function for printing all the elements of a tree in Preoder fashion:**

* In this traversal the root node is visited first and then the left subtree and

Finally the right subtree.

* Preorder traversal for the following binary tree is 1 2 4 5 3 6 7.

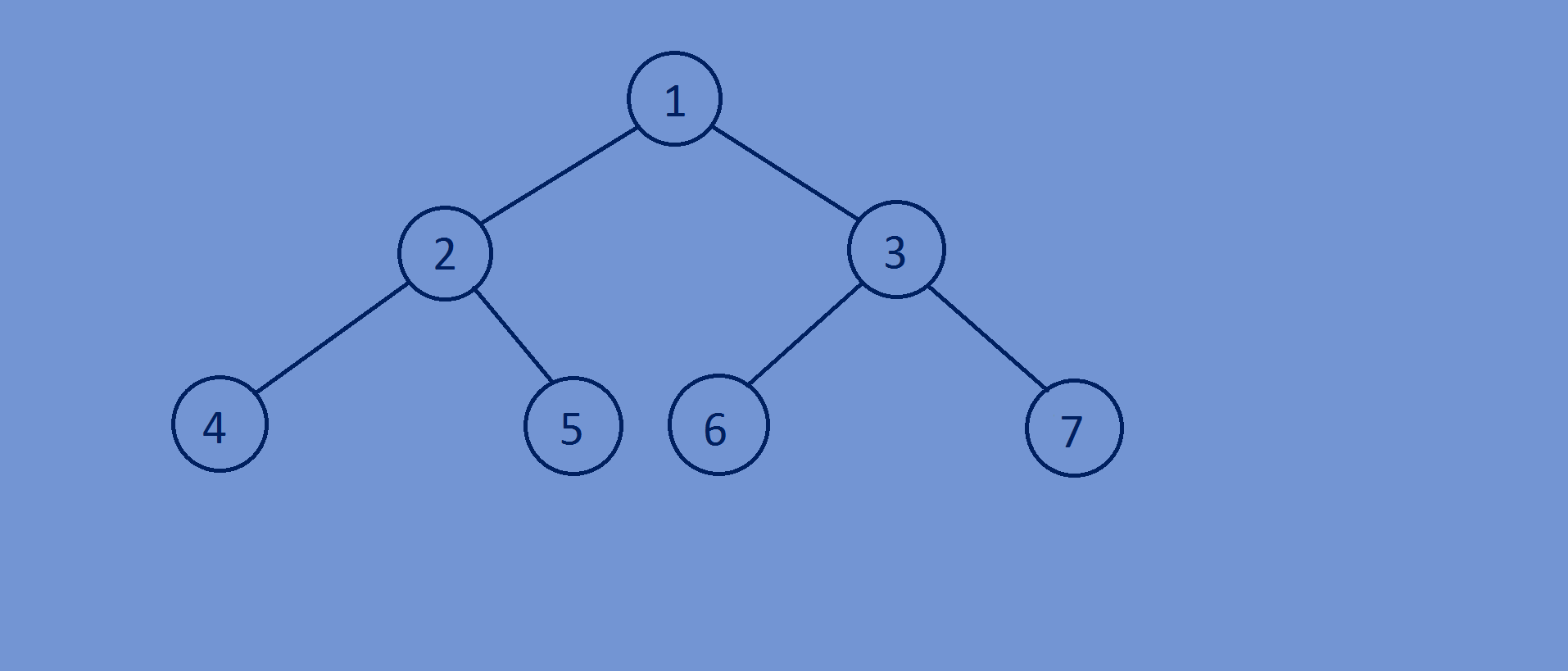
//preorder traversal  
**void** preorderTraversal(Btree \*tree){  
 **if**(tree){  
 printf("%d ",tree->data);  
 preorderTraversal(tree->left);  
 preorderTraversal(tree->right);  
 }  
}

**Function for printing all the elements of a tree in Inorder fashion:**

* In this traversal the left subtree is visited first and then the root node and finally

The Right subtree .

* Inorder traversal for the following binary tree is 4 2 5 1 6 3 7.

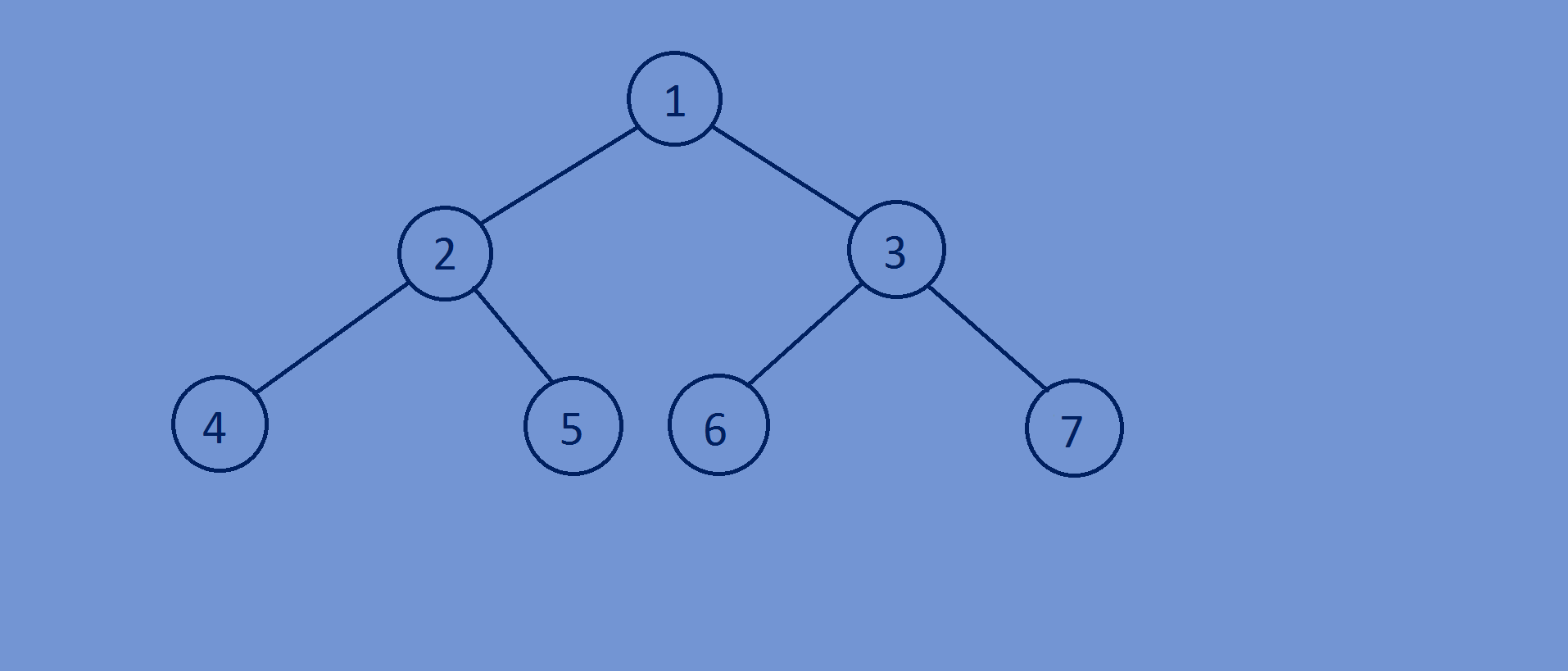
//inorder traversal  
**void** inorderTraversal(Btree \*tree){  
 **if**(tree){  
 inorderTraversal(tree->left);  
 printf("%d ",tree->data);  
 inorderTraversal(tree->right);  
 }  
}

**Function for printing all the elements of a tree in Postorder fashion:**

* In this traversal the left subtree is visited first and then the right subtree and

Finally the root node.

* postorder traversal for the following binary tree is 4 5 2 6 7 3 1.

//postorder traversal  
**void** postorderTraversal(Btree \*tree){  
 **if**(tree){  
 postorderTraversal(tree->left);  
 postorderTraversal(tree->right);  
 printf("%d ",tree->data);  
 }  
}

**Whole program:**

#include<stdio.h>  
#include<stdlib.h>  
  
//creating a Binary tree node.  
**typedef struct** Btree{  
 **int** data;  
 **struct** Btree \*left;  
 **struct** Btree \*right;  
}Btree;  
//create newnode  
Btree \*newnode(**int** data){  
 Btree \*tree=(Btree\*)malloc(**sizeof**(Btree));  
 tree->data=data;  
 tree->left=NULL;  
 tree->right=NULL;  
 **return** tree;  
}  
//preorder traversal  
**void** preorderTraversal(Btree \*tree){  
 **if**(tree){  
 printf("%d ",tree->data);  
 preorderTraversal(tree->left);  
 preorderTraversal(tree->right);  
 }  
}  
//inorder traversal  
**void** inorderTraversal(Btree \*tree){  
 **if**(tree){  
 inorderTraversal(tree->left);  
 printf("%d ",tree->data);  
 inorderTraversal(tree->right);  
 }  
}  
//postorder traversal  
**void** postorderTraversal(Btree \*tree){  
 **if**(tree){  
 postorderTraversal(tree->left);  
 postorderTraversal(tree->right);  
 printf("%d ",tree->data);  
 }  
}  
//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);  
 preorderTraversal(tree);printf("\n");  
 inorderTraversal(tree);printf("\n");  
 postorderTraversal(tree);  
 **return** 0;  
}

**Output:**

1 2 4 5 3 6 7

4 2 5 1 6 3 7

4 5 2 6 7 3 1