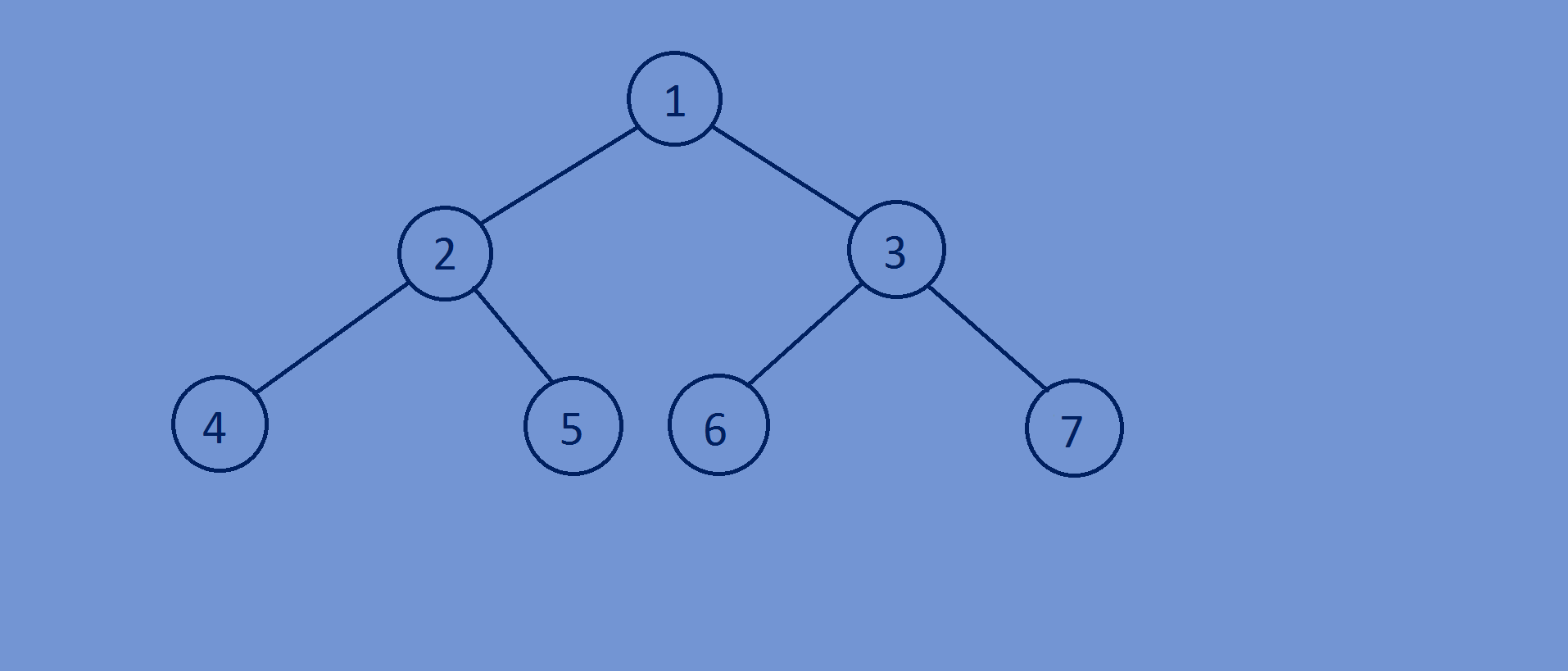
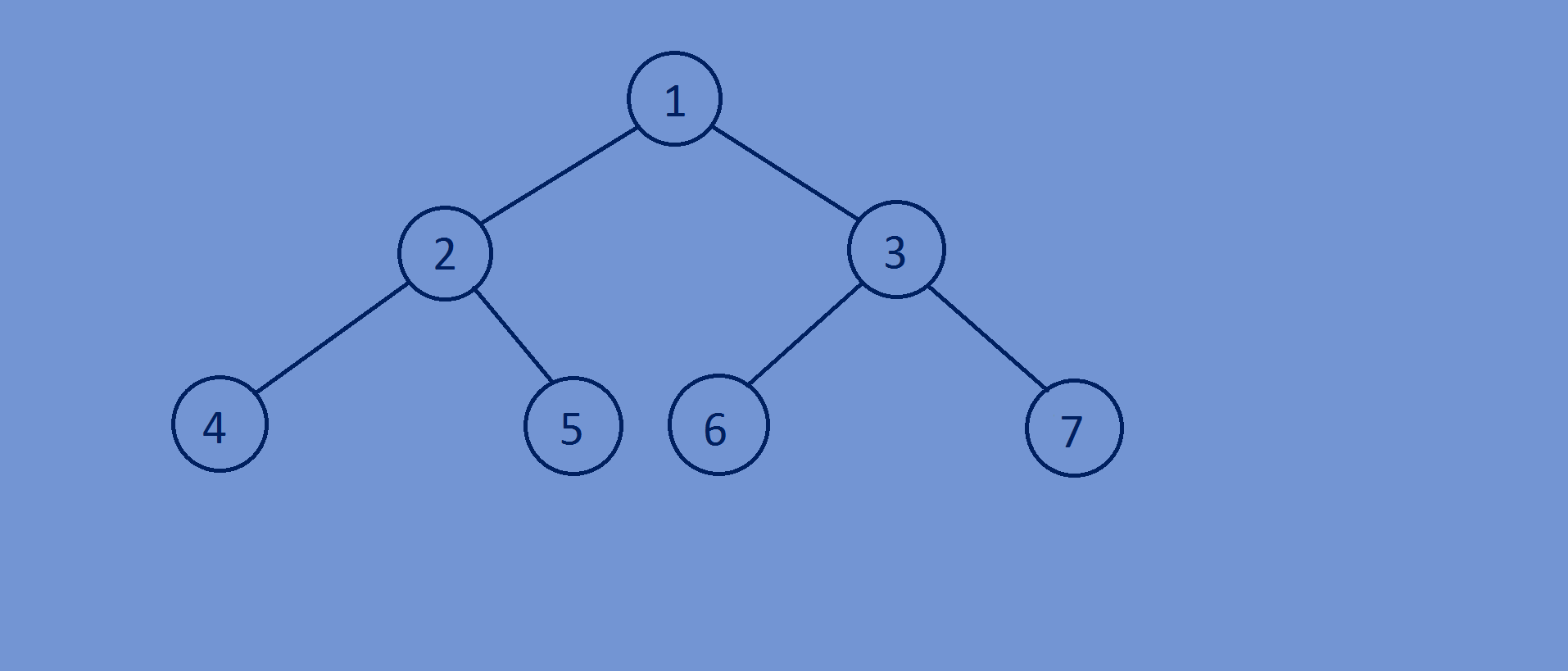
**Data Structures**

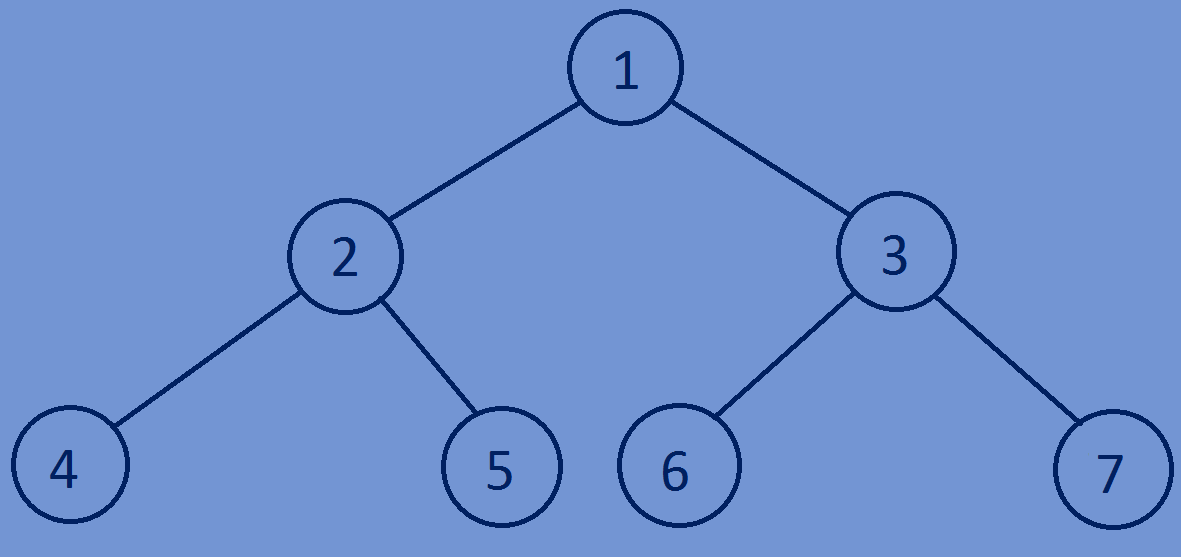
Minimum and Maximum value of a Binary tree

**Finding Minimum and Maximum value of a binary tree:**

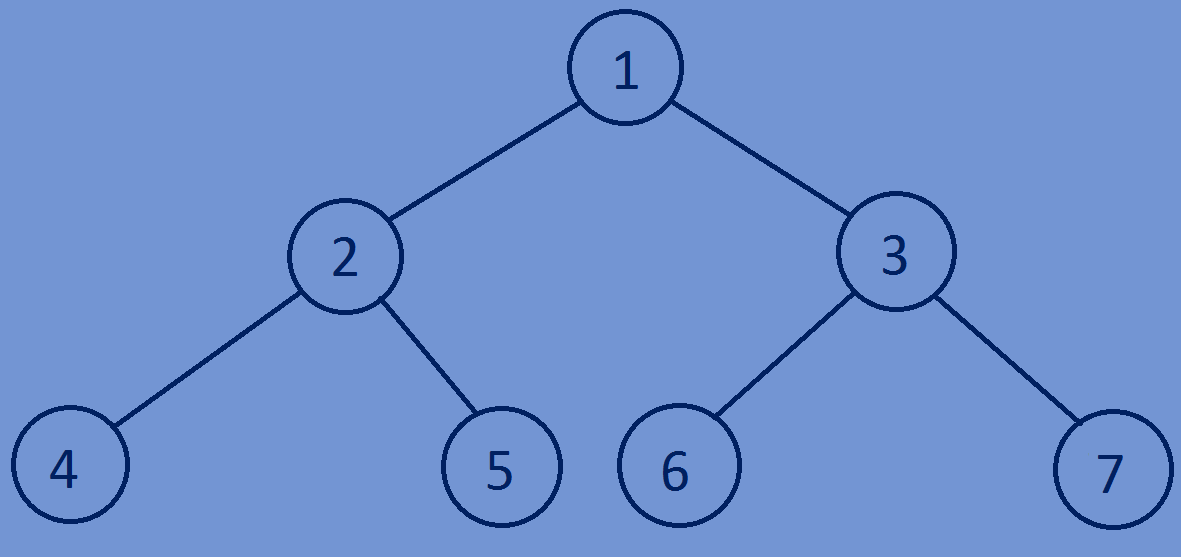
* Return max and min value of the binary tree.
* To find the minimum value,Traverse through the given binary tree and at each node return the minimum of :-
* Node’s data.
* Minimum value of the left subtree of the node.
* Minimum value of the right subtree of the node.
* To find the maximum value,Traverse through the given binary tree and at each node return the maximum of :-
* Node’s data.
* Maximum value of the left subtree of the node.
* Maximum value of the right subtree of the node.



**Function for finding the minimum value of a binary tree:**

//finding min val of a binary tree  
**int** MinValue(Btree \*tree){  
 **if**(tree==NULL){  
 **return** INT\_MAX;  
 }  
 **int** rootval=tree->data;  
 **int** leftval=MinValue(tree->left);  
 **int** rightval=MinValue(tree->right);  
 **if**(leftval<rootval){  
 rootval=leftval;  
 }  
 **if**(rightval<rootval){  
 rootval=rightval;  
 }  
 **return** rootval;  
}

**Function for finding the Maximum value of a binary tree:**

//finding max val of a binary tree  
**int** MaxValue(Btree \*tree){  
 **if**(tree==NULL){  
 **return** INT\_MIN;  
 }  
 **int** rootval=tree->data;  
 **int** leftval=MaxValue(tree->left);  
 **int** rightval=MaxValue(tree->right);  
 **if**(leftval>rootval){  
 rootval=leftval;  
 }  
 **if**(rightval>rootval){   
 rootval=rightval;  
 }  
 **return** rootval;  
}

**Whole program:**

#include<stdio.h>  
#include<stdlib.h>  
#include<limits.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;  
}  
//finding min val of a binary tree  
**int** MinValue(Btree \*tree){  
 **if**(tree==NULL){  
 **return** INT\_MAX;  
 }  
 **int** rootval=tree->data;  
 **int** leftval=MinValue(tree->left);  
 **int** rightval=MinValue(tree->right);  
 **if**(leftval<rootval){  
 rootval=leftval;  
 }  
 **if**(rightval<rootval){  
 rootval=rightval;  
 }  
 **return** rootval;  
}  
//finding max val of a binary tree  
**int** MaxValue(Btree \*tree){  
 **if**(tree==NULL){  
 **return** INT\_MIN;  
 }  
 **int** rootval=tree->data;  
 **int** leftval=MaxValue(tree->left);  
 **int** rightval=MaxValue(tree->right);  
 **if**(leftval>rootval){  
 rootval=leftval;  
 }  
 **if**(rightval>rootval){  
 rootval=rightval;  
 }  
 **return** rootval;  
}  
//main function  
**int** main(){  
 Btree \*tree;  
 tree=newnode(200);  
 tree->left=newnode(2);  
 tree->right=newnode(3);  
 tree->left->left=newnode(4);  
 tree->left->right=newnode(5);  
 tree->right->left=newnode(-2000);  
 tree->right->right=newnode(7);  
 printf("%d\n",MinValue(tree));  
 printf("%d\n",MaxValue(tree));  
 **return** 0;  
}

**Output:**

-2000

200