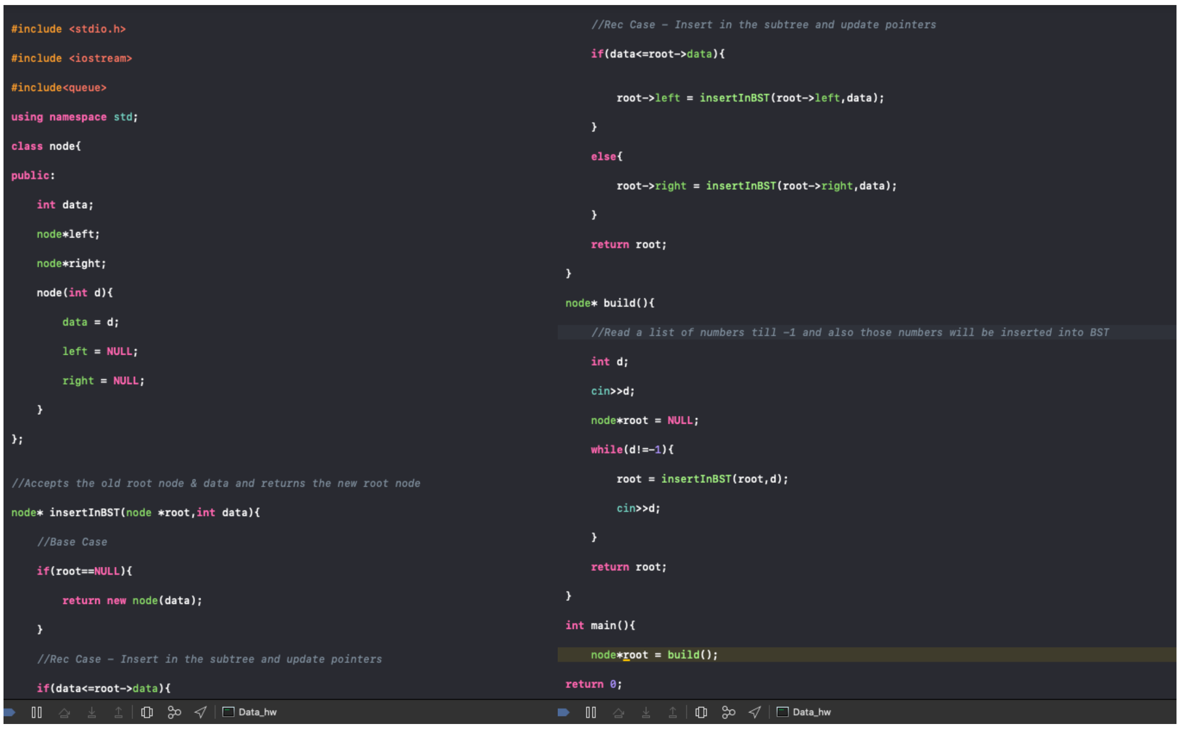
Senanur Güvercinoğlu
150119740

DATA STRUCTURE PROJECT REPORT

A)



```
#include <stdio.h>
#include <iostream>
#include<queue>
using namespace std;
class node{
public:
    int data;
    node*left;
    node*right;
    node(int d){
        data = d;
        left = NULL;
        right = NULL;
    }
};

//Accepts the old root node & data and returns the new root node
node* insertInBST(node *root,int data){
    //Base Case
    if(root==NULL){
        return new node(data);
    }
    //Rec Case - Insert in the subtree and update pointers
    if(data<=root->data){
```

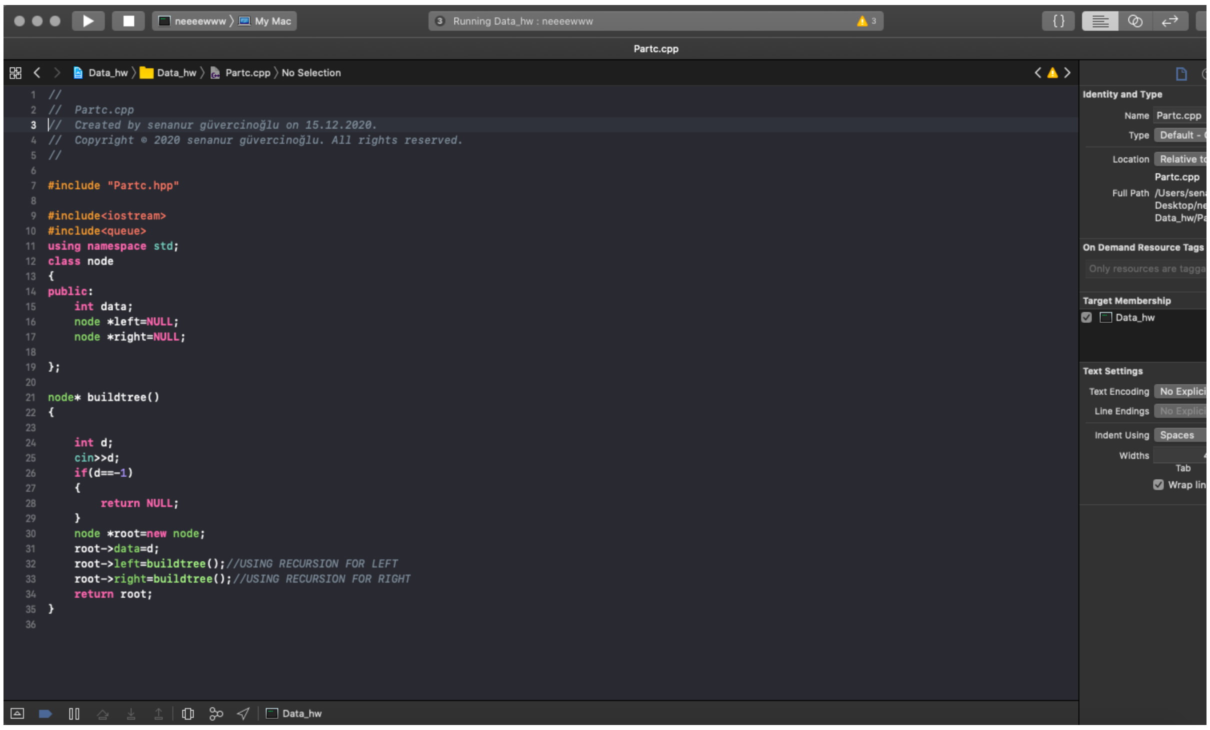
```
    //Rec Case - Insert in the subtree and update pointers
    if(data<=root->data){

        root->left = insertInBST(root->left,data);
    }
    else{
        root->right = insertInBST(root->right,data);
    }
    return root;
}
node* build(){
    //Read a list of numbers till -1 and also those numbers will be inserted into BST
    int d;
    cin>>d;
    node*root = NULL;
    while(d!=-1){
        root = insertInBST(root,d);
        cin>>d;
    }
    return root;
}
int main(){
    node*root = build();
return 0;
```

B) For searching element word, we have to traverse all elements in order . Therefore, searching in binary search tree has worst case complexity of O(n). In general, time complexity is O(h) where **h** is height of tree

Senanur Güvercinoğlu
150119740

C)



```cpp
//
//  Partc.cpp
//  Created by senanur güvercinoğlu on 15.12.2020.
//  Copyright © 2020 senanur güvercinoğlu. All rights reserved.
//

#include "Partc.hpp"

#include<iostream>
#include<queue>
using namespace std;
class node
{
public:
    int data;
    node *left=NULL;
    node *right=NULL;

};

node* buildtree()
{

    int d;
    cin>>d;
    if(d==-1)
    {
        return NULL;
    }
    node *root=new node;
    root->data=d;
    root->left=buildtree();//USING RECURSION FOR LEFT
    root->right=buildtree();//USING RECURSION FOR RIGHT
    return root;
}
```

D) Total access time will be O(n) where n is the number of nodes.

For searching word, we have to traverse all elements (assuming we do breadth first traversal)and it also depends on words frequency. Therefore, searching in binary tree has worst case complexity of O(n).

E) In both tree when we searching or inserting or deleting word, have to traverse all elements in list and also each word have own frequency it means they repeating N times so the time complexity for first tree which is BST => O(h) h is height of BST,Our second tree is Binary Tree In a binary tree, a node can have maximum two children. and the time complexity for our binary tree is O(n).