Problem 1

Assignment 2
Data Structures & Algorithms

Problem Statement

Write a function **bstsort(T)** which outputs the elements of a binary search tree in sorted order. Assume all elements in tree have unique values.

Output

A single line containing **n** (number of nodes in **T**) space separated integers corresponding to the elements of the bst **T** in sorted order

Problem 2

Assignment 2
Data Structures & Algorithms

Problem Statement

Write a function **finddepth(T, P)** which takes a binary tree **T** and a pointer **P** to a node in the tree as inputs and outputs the depth of node pointed to by **P**.

Output

A single integer which is the depth of node pointed to by pointer **P**

Problem 3

Assignment 2
Data Structures & Algorithms

Problem Statement

Write a function **height(T)** to compute the height of a binary tree **T**

Output

A single integer which is the height of tree **T**

Problem 4

Assignment 2
Data Structures & Algorithms

Problem Statement

Write a function **isbst(T)** which takes a binary tree **T** as input and outputs 1 if **T** is a binary search tree and 0 otherwise. Assume integer elements.

Output

A single integer: 1 if **T** is a binary search tree and 0 otherwise.

Problem 5

Assignment 2
Data Structures & Algorithms

Problem Statement

Write a function **randomBST(N)** which takes a positive integer N as input and generates a random binary search tree containing N nodes with distinct integer elements $1, 2, \ldots, N$ (Basically generate a random ordering of integers $1, 2, \ldots, N$ and insert them in the tree). Each time the program is run, it should make a different but valid binary search tree and returns a root pointer to a binary search tree.

Write a function **avgDepth()** which runs the program m=10, 50, 100, 1000 times for N (e.g. N=10, 50, 100, 1000) and prints the average depth for each (m, n) pair.

Generate the 3-D plot of the frequency distribution of average tree depth as function of m and N. What do you observe? Write the observation in the README file.

You can use the following code to plot 3d graph. You need to redirect the output of **avgDepth()** to a $< Outputfile >$ and then run command: python3 $< file.py > < Outputfile >$.

```
import numpy as np
import matplotlib.pyplot as plt
import sys

def read_file(filename):
    depth = []
    with open(filename, "r") as f:
        lines = f.readlines()
        for line in lines:
            depth.append([int(i) for i in line.strip(" \n").split(" ")])
    return np.array(depth)

if __name__ == "__main__":
    m = np.array([10, 50, 100, 1000])
    n = np.array([10, 50, 100, 1000])
    depth = read_file(sys.argv[1])

    X, Y = np.meshgrid(m, n)
    Z = depth.T

    fig = plt.figure()
    ax = plt.axes(projection = '3d')
    # ax.contour3D(X, Y, Z, 50, cmap='binary')
    ax.set_xlabel('m')
    ax.set_ylabel('N')
```

```
ax.set_zlabel('Average Depth')
ax.plot_surface(X, Y, Z, cmap = 'viridis', edgecolor = 'green')
ax.set_title('3D plot of Average Depth')
plt.show()
```

The function to be implemented are:-

1. Node* randomBST(int N);
2. void avgDepth();

Output

The function **randomBST(N)** returns a root pointer to a binary search tree.

The function **avgDepth()** prints 2-d array indicating average depth of tree for that (m, n) pair. The format of output should be: Print 4 lines for each m = 10, 50, 100, 1000 in this order, in each line print 4 space separated numbers for n = 10, 50, 100, 1000 in this order representing average depth of the tree for that (m, n).

Problem 6

Assignment 2
Data Structures & Algorithms

Problem Statement

Write a function **inRange(T, k1, k2)** which takes three inputs - a root pointer to a binary search tree **T** and two integers **k1** and **k2** where **k1** $\leq$ **k2**. As output, it prints all the nodes **X** in the tree such that **k1** $\leq$ **X** $\leq$ **k2**.

The function to be implemented is:-

1. void inRange(Node* root, int k1, int k2);

Output

Print all the nodes **X** in the tree such that **k1** $\leq$ **X** $\leq$ **k2** separated by a space.

Optional

Assignment 2
Data Structures & Algorithms

Problem Statement

(not graded.) Modify the following code ASCII Binary Tree to visualize binary trees in the assignment.

Instructions

Assignment 2
Data Structures & Algorithms

- You need to make three files - **main.c**, **bst.c** and **bst.h**.
- Add all the functions in bst.c file.
- All the functions' declarations and structs are to be added in bst.h file.
- You need to use main.c file to call functions implemented in bst.c file.
- Add the following **Node** struct in bst.h file. Use this struct for creating the node of the tree. The Node struct contains data variable to store the value of the node and two pointers for left and right child. You need to use this struct to create the node of the tree for every problem. You can add other variables in this **Node** struct.

```
typedef struct Node
{
    int data;
    int height;
    struct Node* left;
    struct Node* right;
}Node;
```

- Make a **README** file and add very short explanation of your code and time complexity of the code. Add the observation of graph obtained in Problem 5 in README.
- Put **bst.c**, **bst.h**, **main.c** and **README.md** file in a directory named $< RollNumber >$. Zip the directory with name $< RollNumber >$.zip and upload on moodle.