PROG20799: Data Structures and Algorithms in C

Evaluation: **10 points, 7.5%** of your final grade.
Due date: See SLATE.
Late submission: **10% per day** penalty, up to 3 days.

In this assignment you need to create a **binary search tree (BST)** to store student's data. Your goal is to collect students' info (ID and GPA), print records and modify them if needed.
You must use the program template **main.txt** file and function prototypes given to you.
Please check the demo version: https://youtu.be/r3oy6_1e-M8

You must use a **BST** with the following **structure**:

```
typedef struct node BST_node;
struct node {
   short id;
   float gpa;
   BST_node *leftChild;
   BST_node *rightChild;
};
```

Requirements:

- Please download and use **main.txt** template file.
  You must only use this template file with function prototypes or you'll get a **zero grade**.
- You absolutely must use the structure mentioned above.
  You'll get a **zero grade** if you use anything else to store student's info.
- Structures must be allocated on the **HEAP**. You'll get a **minimal grade** otherwise.
- You must use **fgets()** to get both the **id** and **gpa** of the student (use **MAX_LEN** macros).
- You must allocate student's name on the HEAP and use the **strto…()** function to convert input to the appropriate numerical value. Do NOT use **scanf(),** or **atof()** anywhere!
- You must use **BST based on ID.** I.e. for input with id 2000, 1000 and 3000 you'll have root with id 2000, left node with id 1000 and right node with ID 3000.
- You must implement and use all the functions listed in the template file.
- You cannot modify the **main()** function given to you in the template file.
- You must destroy the **BST** and deallocate memory at the end of the program.
  Please use function **removeBST(..)** accordingly.
- Your program must work similarly to the Demo version posted on YouTube.
  Namely, you must ignore the incorrect input, have same error notifications, check for duplicate ids, display the BST correctly (minimum ID first and maximum ID last).
- Your solution should have optimal time and space complexity with no repetitive code.
  You could/should add your own function(s) if needed. For instance, to avoid repetitive code you should implement **get_id()** function that checks the id during input. See Hints below.

Hints:

1. Check in-class exercise(s) to implement all the functions you need.

2. Notice that when you display BST, the record with **minimum ID** always displayed first and the record with **maximum ID** always displayed last.
   Which of the traversal functions (preorder, postorder, inorder) can help you?

3. When you enter **empty id** it is assumed there are no more nodes to add to the list.

4. In the **removeBST()** function you need to de-allocate memory for all the nodes.
   To visit the nodes use same algorithm as in the traversal functions (preorder, postorder, etc).

5. Changing ID of a node can/should be done inside **move_node()** function.
   First, you'd need to detach the node from the BST and then insert it again with a new ID.
   Check in-class exercise(s) how to remove nodes in the BST.

6. To avoid repetitive code you should implement **get_id(BST_node*, int flag)** function that checks for correct **ID** input from console (between 1000 and 9999), checks for duplicates (flag=1), and returns value -1 if **ID** is incorrect, -2 if **ID** input is empty or the actual **ID** value.
   This function can be used in both **createBST_node()** and **changeBST()** functions.

Submission:

This is an individual assignment. Even partially copied code will be subject to regulations against academic integrity. Do NOT discuss or share your solution with anybody. Posting this assignment or solution on the Internet is a violation of the Student Code of Conduct.

- Please make sure your code is POSIX-compliant (works in Cygwin)!
- Your submission must be one **text file only**: **assignment4.txt**
- Please copy **main.c** as text file **assignment4.txt** with extension **.txt**
- You must upload **assignment4.txt** file to the Assignment Dropbox by due date.
- Verify that **assignment4.txt** file you submitted can be opened by Notepad in Windows.
- You'll get **-3 points** if you submit incorrect file (.zip file, or file with incorrect name/extension).
- Your submission must be unique or have references.
- Please **self-evaluate your code** in the comments section of the Dropbox.
- **Late submissions are penalized 10% / day (1 point / day), up to -3 points.**
  Submissions won't be accepted after 3 days.

Grading Scheme:

- See **Main requirements** and also **Course_Introduction.pdf**.
  Deductions will be applied if partial functionality is provided.
  You can get up to 70% if **changeBST()** not implemented (but everything else is perfect!)
- You'll get **zero grade** if your code doesn't compile.
  Any compilation warning is a major mistake.
- Comments are not required but recommended. However, "debugging code" or commented out "old code" is a minor mistake (unless it's clearly stated in the comments that you want me to take a look at such code).
- Please check submission requirements for any additional deductions.