CSCI 2520 Data Structures and Applications
Assignment 3
Deadline: 23:55, APR. 28, 2020
Total Marks: 100

Submission:
In this Assignment, you need to answer questions 1 and 2 in one pdf file. For questions 3, 4, 5, you need to provide .c file for each question, which can be compiled and give the correct answer, the name should be q3.c, q4.c, and q5.c respectively.
Then compress all 4 files (one pdf file for q1, q2 and three .c files for q3, q4 and q5 respectively) as one zip named as `your_student_id_assign3.zip` and submit it via Blackboard.

1. Complexity Analysis (20 marks)

Analyze the time complexity of the following functions and give some explain briefly. Please use the Big-O Notation. (note that the time complexity should be as tight as possible)

Question 1:

```
void function(int x) {
      int sum = 0;
      for (int i = 1; i <= x; i = i * 2) {
            for (int j = 1; j <= x / i; ++j) {
                  sum += j;
            }
      }
}
```

Question 2:

```
long long function(long long a,  long long b) {
      if (b == 0) return 1;
      long long res = function(a, b / 2);
      if (b % 2)
            return res * res * a;
      else
            return res * res;
}
```

2. Build BST from different orders (20 marks)

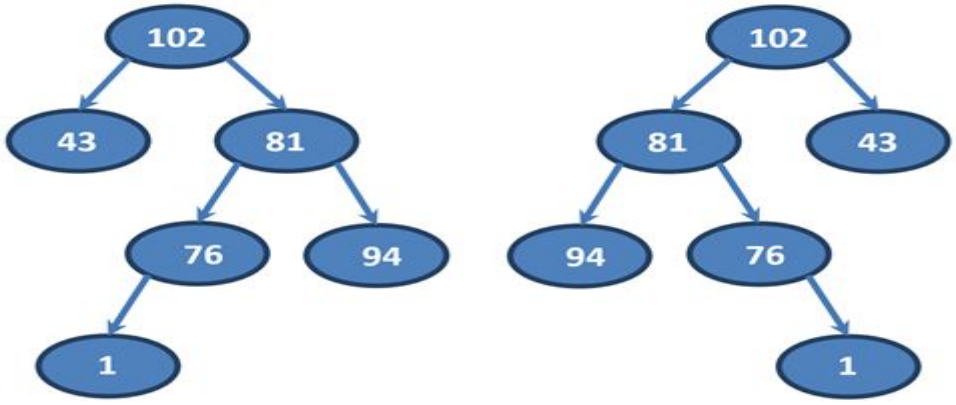
(1) Given *inorder* and *postorder* traversal of a tree, Can you construct a certain binary tree? Why?

(2) Given *preorder* and *postorder* traversal of a tree, Can you construct a certain binary tree? Why?

3. Mirror the BST (20 marks)

Given a binary tree. You need to do the mirror operator and get the tree which is symmetric of the input tree.

For example:



Input **Output**

Format of functions:

```
bt* Mirror(bt* tree);
```

The Definition for binary tree is below:

```
typedef struct btCDT bt;
struct btCDT {
      int val;
      bt *left;
      bt *right;
};
```

Sample program is provided in **assign3-3.c**, you just need to finish the implementation of function **`Mirror`** in the tail of the file. (You can add extra functions to help you finish your codes).

4. Find the closest value (20 marks)

Given a binary search tree and a number, find the closest value (the difference between the given number and the answer is the smallest) in the BST.

If there are many answers, just return the smallest one.

Format of functions:

```
int Find_closest(bst* tree, int num);
```

The Definition for binary search tree is below:

```
typedef struct bstCDT bst;
struct bstCDT {
      int val;
      bst *left;
      bst *right;
};
```

Sample program is provided in **assign3-4.c**, you can find an example in the file. You just need to finish the implementation of function **`Find_closest`** in the tail of the file. (You can add extra functions to help you finish your codes).

5. Rebuild the BST (20 marks)

After several insertions, you may have an unbalanced binary search tree. So now give you a binary search tree, you need to rebuild it and make it balanced.

Format of functions:

```
bst* Rebuild(bst* tree);
```

The Definition for binary search tree is below:

```
typedef struct bstCDT bst;
struct bstCDT {
      int val;
      bst *left;
      bst *right;
      int height;
};
```

Sample program is provided in **assign3-5.c**, you just need to finish the implementation of function `Rebuild` in the tail of the file.(You can add extra functions to help you finish your codes).