# EEEC10008(515169) S23: Homework 1

## Due: 2023/4/18(Tues.) 23:59

**[ Instruction ]**

- Please put your source code files of each problem into separate folder named StudentID_hw1_1 and StudentID_hw1_2, compress these two folders to **zip** files separately (Ex: 111511000_hw1_1.zip, 111511000_hw1_2.zip), and upload these two zip files to e3 before deadline.
- **If zipped file' s or source code file's name is wrong, your score of this homework is 30% off.**
- **Your source code files should be able to be compiled and executed on our server.**
- Your output should follow the example, otherwise you will not get full credit.
- If you have any question, please send an email to TA or leave a message in the line account.
- If necessary, you can get the files provided in each problem from `/home/share/hw1/`.

**[ Problem 1 ]: Binary Search Tree**

In computer science, a **binary search tree** (**BST**), also called an **ordered** or **sorted binary tree**, is a rooted binary tree data structure with the key of each internal node being greater than all the keys in the respective node's left subtree and less than the ones in its right subtree. The time complexity of operations on the binary search tree is directly proportional to the height of the tree. In this problem, you have to implement a class of Binary Search Tree (`class BST`).

- `class BST` function members
  - Insert node to BST: `BST_Insert(void* dataPtr)`
  - Delete node from BST: `bool BST_Delete(void* dltKey)`
  - Traverse BST: `BST_Traverse(void (*process) (void* dataPtr))`
  - Check BST is full or not (by declaring a new dynamic `node`): `BST_Full()`
  - Check BST is empty or not: `BST_Empty()`
  - Return the number of node in BST: `BST_Count()`
- Please write the class declaration in BST.h, write the definition of the class member in BST.cpp and execute the program in hw1-1.cpp.
  - In the hw1-1.cpp, you should follow the example format and add your own code
  - `int compareInt(void* num1, void* num2)`: integer compare function, note the data type. Should be pointed by the BST class member `int compareInt(void* num1, void* num2)` by passing it into the BST constructor.

- `void printBST(void* num1)`: should be passed to the `BST_Traverse(void (*process) (void* dataPtr))` for printing out the data
- `void printMenu()`: print the option menu

`hw1-1.cpp` is shown as below:

```
// BST.h
#include "BST.h"
using namespace std;
int compareInt(void* num1, void* num2);
void printBST(void* num1);
void printMenu();

int main(){
    //Write your own code here

    return 0;
}
int compareInt(void* num1, void* num2){
    //Write your own code here
}
void printBST(void* num1){
    //Write your own code here
void printMenu(){
    //Write your own code here
}
```

- Hint for implementing the functions in the hw1-1.cpp
  - 1. Insert: `BST_Insert(void* dataPtr)`
  - 2. Delete: `BST_Delete(void* dltKey)`, `BST_Empty()`, `BST_Traverse(void (*process) (void* dataPtr))`
  - 3. Print the BST: `BST_Empty()`, `BST_Traverse(void (*process) (void* dataPtr))`
  - 4. Check the BST is full or not: `BST_Full()`
  - 5. Print the number of nodes in BST: `BST_Count()`

Option menu is shown below

```
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter:
```

- In the BST.h, you should also add new function members with the original example format fixed. Namely, follow the example format and add function members.

`BST.h` is shown as below:

```
// BST.h
#ifndef BST_H
#define BST_H
#include <iostream>
using namespace std;

struct node{
    void* dataPtr;
    node* left;
    node* right;
};
class BST{
private:
    int count;
    int (*compare) (void* argu1, void* argu2);
    node* root;
public:
    BST(int (*compare) (void* argu1, void* argu2));
    ~BST();
    bool BST_Insert(void* dataPtr);
    bool BST_Delete(void* dltKey);
    void BST_Traverse(void (*process) (void* dataPtr));
    bool BST_Empty();
```

```
    bool BST_Full();
    int BST_Count();

    node* _insert(node* root, node* newPtr); //Call by BST_Insert
    //Add more function members here
};

#endif
```

The example output is shown as below:

```
$ ./hw1-1
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 3
The BST is empty.
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 1
Enter an interger: 6
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 1
Enter an interger: -9
The following are six options for your BST:
1. Insert
```

```
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 1
Enter an interger: 15
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 1
Enter an interger: -4
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 1
Enter an interger: 53
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 1
Enter an interger: 13
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
```

```
Please enter: 3
-9 -4 6 13 15 53
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 4
BST is not full yet
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 5
There are 6 nodes in BST.
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 2
Enter an interger: 7
The number is not in BST.
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 2
Enter an interger: 13
New BST: -9 -4 6 15 53
The following are six options for your BST:
```

```
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 2
Enter an interger: 15
New BST: -9 -4 6 53
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 2
Enter an interger: -4
New BST: -9 6 53
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 2
Enter an interger: -9
New BST: 6 53
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 2
Enter an interger: 6
New BST: 53
The following are six options for your BST:
1. Insert
```

```
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 2
Enter an interger: 53
The BST is empty.
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 7
The following are six options for your BST:
1. Insert
2. Delete
3. Print the BST
4. Check the BST is full or not
5. Print the number of nodes in BST
6. Exit
Please enter: 6
<End of Program>
```

**[ Problem 2 ]: Stock System**

Please complete the definition of the class `Stock` to support the correct execution of the main program (written in `hw1-2.cpp`).

You should write the `Stock.cpp` and add proper codes in `Stock.h`. You only need to put these three files in the folder StudentID_hw1_2 and zip it to StudentID_hw1_2.zip.

Do not modify `hw1-2.cpp`.

You can only add codes in `Stock.h` but do not modify the existed codes.

Execute: ./hw1-2

Define a class called Stock that has member variables to record who owns the stock(string), how many he owns(int), and the average price(double). There are also four static variables, initial price, limit ratio, current price, and trade available.

- Initial price = 57.88
- Limit ratio = 0.1
- Current price will change after every trading action.
  Price(new) = Price(old) + rand() % 500 / 100 – 2.5
- trade available
  Once the difference ratio is bigger than limit ratio, the trade will become not available any more.

There are two overloading operator(+, -). For example, there are two Stock variables, S1 and S2. S1 + S2 represents that

- S1.avg_price=(S1.avg_price * S1.number + S2.avg_price * S2.number)/(S1.number+S2.numer)
- S1.number += S2.number

There are three trading actions.

1. Buy
   - Check trade is available
   - Ask the number of trading
   - Use the operator +
2. Sell
   - Check trade is available
   - Ask the number of trading
   - Check the owner has enough number of stock
   - Use the operator -
3. Show stock
   - Shown in the output sample

`Stock.h` is shown as below:

```
// Stock.h
#ifndef _STOCK_H_
#define _STOCK_H_
#include <string>

using namespace std;

class Stock
{
private:
    static const double init_price;
    static const double limit_ratio;
    static double cur_price;
    static bool trade_available;
    int ticket_num;
    double avg_buy_price;
    //double total;
    const string owner_name;

public:
    Stock();
    Stock(string name);
    Stock(string name, int n1);
    //Stock(string name, double total1);

    void operator + (const Stock &);
    void operator - (const Stock &);

    void buy();
    void sell();
    void show_stock();

    string get_owner_name() const;

    static void show_current_price();
    static void check_trade_is_available();
    static void refresh_current_price();

};
#endif
```

`hw1-2.cpp` is shown as below:

```
//hw1-2.cpp
#include <iostream>
#include <string>
#include <vector>
#include <cstdlib>
#include <time.h>
#include "stock.h"

using namespace std;

int main()
{
    srand(531);
    int n;
    cout << "How many people want to buy this stock: ";
    cin >> n;
    vector<Stock> stock_vec;
    cout << "These people's name are: ";
    string temp;
    for(int i=0; i<n; i++){
        cin >> temp;
        stock_vec.push_back(Stock(temp));
    }

    bool first = true;

    while(true){
        if(!first){
            Stock::refresh_current_price();
            Stock::check_trade_is_available();
        }
        else{
            Stock::show_current_price();
        }
        first = false;

        bool flag = true;
        while(flag){
            cout << "Who want to do the trade: ";
            cin >> temp;
```

```
            for(int i=0; i<n; i++){
                if(stock_vec[i].get_owner_name() == temp){
                    flag = false;
                    temp = "4";
                    while(temp != "1" && temp != "2" && temp != 
"3"){
                        cout << "1.buy the stock / 2.sell the stock / 
3.show stock: ";
                        cin >> temp;
                        if(temp == "1"){
                            stock_vec[i].buy();
                        }
                        else if(temp == "2"){
                            stock_vec[i].sell();
                        }
                        else if(temp == "3"){
                            stock_vec[i].show_stock();
                        }
                        else{
                            cout << "wrong choice. please enter 
again." << endl;
                        }
                    }
                }
            }
            if(flag){
                cout << "No this person. please enter again" << 
endl;
            }
        }
    }
    return 0;
}
```

Execution result is shown as below:

```
$ ./hw1-2
```

Output Sample:

```
How many people want to buy this stock: 3
These people's name are: su wu wang
----------------------------------------
current price of the stock: 57.88
Who want to do the trade: su
1.buy the stock / 2.sell the stock / 3.show stock: 3
su, you have 0 stocks and the average price is 0 //show_stock()
----------------------------------------
current price of the stock: 56.18
Who want to do the trade: wi
No this person. please enter again
Who want to do the trade: wu
1.buy the stock / 2.sell the stock / 3.show stock: 1
How many tickets do you want to buy: 10
----------------------------------------
current price of the stock: 56.45
Who want to do the trade: wu
1.buy the stock / 2.sell the stock / 3.show stock: 4
wrong choice. please enter again.
1.buy the stock / 2.sell the stock / 3.show stock: 3
wu, you have 10 stocks and the average price is 56.18
----------------------------------------
current price of the stock: 56.21
Who want to do the trade: wu
1.buy the stock / 2.sell the stock / 3.show stock: 2
How many tickets do you want to sell: 3
----------------------------------------
current price of the stock: 55.8
Who want to do the trade: wu
1.buy the stock / 2.sell the stock / 3.show stock: 3
wu, you have 7 stocks and the average price is 56.1671
----------------------------------------
current price of the stock: 54.06
Who want to do the trade: wang
1.buy the stock / 2.sell the stock / 3.show stock: 1
How many tickets do you want to buy: 50
----------------------------------------
```

```
current price of the stock: 53.36
Who want to do the trade: wang
1.buy the stock / 2.sell the stock / 3.show stock: 1
How many tickets do you want to buy: 100
-----------------------------------------
current price of the stock: 51.92
trade is not available anymore
Who want to do the trade: wang
1.buy the stock / 2.sell the stock / 3.show stock: 1
The trade is not available today.
-----------------------------------------
current price of the stock: 50.29
trade is not available anymore
Who want to do the trade: wang
1.buy the stock / 2.sell the stock / 3.show stock: 2
The trade is not available today.
-----------------------------------------
current price of the stock: 51.22
trade is not available anymore
Who want to do the trade: wang
1.buy the stock / 2.sell the stock / 3.show stock: 3
wang, you have 150 stocks and the average price is 53.5933
-----------------------------------------
current price of the stock: 51.49
trade is not available anymore
Who want to do the trade: su
1.buy the stock / 2.sell the stock / 3.show stock: 1
The trade is not available today.
-----------------------------------------
current price of the stock: 52.38
trade is not available anymore
Who want to do the trade: wu
1.buy the stock / 2.sell the stock / 3.show stock: 1
The trade is not available today.
-----------------------------------------
current price of the stock: 50.22
trade is not available anymore
Who want to do the trade: ^C
```