Lab 6 (Practice)
Class and Objects

Question-1.

Now, consider a program to find whether a number exists in an array. The simple idea is to go through the whole array to check whether any elements equal to the number. However, if the array is sorted, we have better approach to solve the problem called binary search.

For example:
Assume that we know have a sorted array in ascending order, and want to check whether a number n exists in the array or not.

1. We keep three indices low, mid, and high. Initially, we assign 0 to low, as well as the length of the array to high.
2. Then we calculate the average of the low and high, and store the result in mid. Then compare the element at the position of mid in the array with n.
    a. If they are equal, we get the answer.;
    b. If n is larger, we can know the possible range of n must be the interval from mid+1 to high, as the array is sorted, and we assign the value of mid+1 to low;
    c. If n is smaller, the range must be the interval from low to mid-1, and we assign the value of mid-1 to high.
3. We repeat step 2 until we find the position of n, or high is larger than low, which means n does not exist in the array.

Now we implement the binary search on the record class in the basic question 3. The search will find and output all the records with a certain balance. If there are multiply balance with the searching value, output them in ascending order according to the time.

Expected Outcomes

```
Example 1:
Enter the number of the records:
10
Enter the contents of each records:
5 20 400
3 12 -65.3
5 11 -9.4
5 11 -67.3
10 31 200
6 2 -230.5
2 11 -67.3
5 25 -127.5
7 2 200
8 29 -67.3
Enter the balance to find:
-67.3
3 records found!
2 11 -67.3
5 11 -67.3
8 29 -67.3
```

```
Example 2:
Enter the number of the records:
10
Enter the contents of each records:
5 20 400
3 12 -65.3
5 11 -9.4
5 11 -67.3
10 31 200
6 2 -230.5
2 11 -67.3
5 25 -127.5
7 2 200
8 29 -67.3
Enter the balance to find:
200
2 records found!
7 2 200
10 31 200
```

Question-2

Design a class (named Lock) to implement the function of a security lock. The lock should implement the following operations:

a. If the current operation is to *(turn left, n steps),* then the counter for Left is incremented by n. In addition, if the previous operation is to *turn right*, the counter for Right is reset to 0.

   For example, the previous operation is (turn right, 5 steps) and the counter for Right has been incremented by 5. If the current operation is (turn left, 2 steps), the counter for Left will be incremented by 2 and the counter for the Right will be set to 0.

b. If the current operation is to *(turn right, n steps),* then the counter for Right is incremented by n. In addition, if the previous operation is to *turn left*, the counter for Left is reset to 0 too.

c. If the above two counter values are the same as the default setting, when pressing the *unlock* button, then the lock is successfully unlocked. The user can hear a "click" sound (modeled as a text message in this exercise).

d. The lock has a function *reset* to set the values of left and right counters to 0.

e. If the current operation is *end*, the program will exit.

Notes:

1. All the members should be private.
2. There is only one constructor for the class (i.e. Lock(int left, int right)) for the user to set the values for the two counters to unlock the lock. Due to security reasons, the two values cannot be changed any more after initialization.
3. There two ways to terminate the program: one is the successful unlock, the other is enter the terminate operation to the program. For convenience, 'r' represents turn right, 'l' represents turn left, 'u' represents unlock during operations, 's' represents reset the counters, and 'e' represents end the program.

Expected Output

```
Example 1:
Enter the value for the left counter:
3
Enter the value for the right counter:
2
Enter the operation:
r 2
Enter the operation:
l 3
Enter the operation:
u
Failed!
Enter the operation:
r 2
Enter the operation:
u
Successful!
```

```
Example 2:
Enter the value for the left counter:
3
Enter the value for the right counter:
2
Enter the operation:
e
Program terminate!
```

```
Example 3:
Enter the value for the left counter:
3
Enter the value for the right counter:
2
Enter the operation:
r 2
Enter the operation:
u
Failed!
Enter the operation:
l 5
Enter the operation:
u
Failed!
Enter the operation:
s
Enter the operation:
r 2
Enter the operation:
u
Failed!
Enter the operation:
l 3
Enter the operation:
u
Successful!
```