Object-oriented Programming Homework1 Questions

YoungWoon Cha
CSE Department
Spring 2023

KU KONKUK UNIVERSITY

Homework 1 Question 1

Q1 *Fraction Class Part 1*

A *fraction* is a ratio of two integers such as 3/4, 1/2, 7/5, and so on.

There is no built-in type in C++ that can represent a fraction; we need to create a new type for it with two data members of type integer.

The first we call the *numer* (abbreviation for *numerator*); the second the *denom* (abbreviation for *denominator*).

Q2 *Fraction Class Part 2*

The invariants that we need to worry about in the fraction objects are three conditions:

- **The numerator and the denominator should not have a common factor. e.g. 6/9 should be reduced to 2/3.**
- **The denominator cannot be 0. A fraction such 2/0 is undefined.**
- **The sign of the fraction, is the product of the sign of the numerator and denominator and should be set as the sign of the numerator. e.g. 2/-3 → -2/3, -2/-3 → 2/3**

The *gcd* function finds the greatest common divisor between the numerator and denominator.

The normalized function takes care of the three invariants using the *gcd* function.

Question 1

- Given the interface and the application file, implement Fraction class so that you can get the same result as in the output.

```
c++ -c fraction.cpp                    // Compilation of implementation file
c++ -c app.cpp                         // Compilation of application file
c++ -o application fraction.o app.o    // Linking of two compiled object files
Application                            // Running the executable file
```

```
Run:
Printing four fractions after constructed:
fract1: 0/1
fract2: 2/3
fract3: -11/8
fract4: -11/8
Changing the first two fractions and printing them:
fract1: 4/1
fract2:-2/5
Testing the changes in two fractions:
Numerator of fract1: 4
Denominator of fract2: 5
```

Q1 Interface File Part 1

The fraction.h file

```
1  /*************************************************************
2   * The interface file fraction.h defining the class Fraction   *
3  /*************************************************************
4  #include <iostream>
5  using namespace std;
6
7  #ifndef FRACTION_H
8  #define FRACTION_H
9
10 class Fraction
11 {
12      // Data members
13     private:
14         int numer;
15         int denom;
16
17     // Public member functions
18     public:
19         // Constructors
20         Fraction (int num, int den);
```

Q1 Interface File Part 2

The fraction.h file

```
21          Fraction ();
22          Fraction (const Fraction& fract);
23          ~Fraction ();
24          // Accessors
25          int getNumer () const;
26          int getDenom () const;
27          void print () const;
28          // Mutators
29          void setNumer (int num);
30          void setDenom (int den);
31
32      // Helping private member functions
33      private:
34          void normalize ();
35          int gcd (int n, int m);
36  };
37  #endif
```

Q1 Application File Part 1

Application file (app.cpp)

```
1  /*************************************************************
2   * The application file app.cpp uses the Fraction objects.     *
3   *************************************************************/
4  #include "fraction.h"
5  #include <iostream>
6  using namespace std;
7
8  int main ( )
9  {
10      // Instantiation of some objects
11      Fraction fract1 ;
12      Fraction fract2 (14, 21);
13      Fraction fract3 (11, -8);
14      Fraction fract4 (fract3);
15      // Printing the object
16      cout << "Printing four fractions after constructed: " << endl;
17      cout << "fract1: ";
18      fract1.print();
19      cout << "fract2: ";
20      fract2.print();
```

Q1 Application File Part 2

Application file (app.cpp)

```
21      cout << "fract3: ";
22      fract3.print();
23      cout << "fract4: ";
24      fract4.print();
25      // Using mutators
26      cout << "Changing the first two fractions and printing them:" << endl;
27      fract1.setNumer(4);
28      cout << "fract1: ";
29      fract1.print();
30      fract2.setDenom(-5);
31      cout << "fract2: ";
32      fract2.print();
33      // Using accessors
34      cout << "Testing the changes in two fractions:" << endl;
35      cout << "fract1 numerator: " << fract1.getNumer() << endl;
36      cout << "fract2 numerator: " << fract2.getDenom() << endl;
37      return 0;
38  }
```

KU KONKUK UNIVERSITY

Homework 1 Question 2

Q2 *Time Class*

The time class has three data members:
(*hours) (minutes) (seconds)*.

We will have a parameter constructor and a default constructor, but we do not need a copy constructor.
We have three accessor functions to get hours, minutes, and seconds respectively.

We use only one mutator function, which we call *tick* that, each time it is called, moves the time object one second forward.

We need to worry about two conditions:

- **All three data members need to be non-negative; otherwise, the program is aborted. We can not have a negative time.**
- **The hours should be between 0 to 23, the minutes should be between 0 and 59, and the seconds should also be between 0 and 59. We use modulo arithmetic to keep these values in the range.**

Question 2

- Given the interface and the application file, implement Fraction class so that you can get the same result as in the output.

```
c++ -c time.cpp        // Compilation of implementation file
c++ -c app.cpp         // Compilation of application file
c++ -o application time.o app.o       // Linking of two object files
application                 // Running the executable file
```

```
Run:
Original time: 4:5:27
Time after 143500 ticks: 19:57:7
```

Q2 Interface File Part 1

The interface file (time.h)

```
1  /*************************************************************
2   * The interface file for time.h class                       *
3   *************************************************************/
4  #include <iostream>
5  using namespace std;
6
7  #ifndef TIME_H
8  #define TIME_H
9
10 class Time
11 {
12      private:
13          int hours;
14          int minutes;
15          int seconds;
16      public:
17          Time (int hours, int minutes, int seconds);
18          Time ();
19          ~Time ();
20          void print() const;
```

Q2 Interface File Part 2

The interface file (time.h)

```
21          void tick();
22
23      private:
24          void normalize (); // Helping function
25  };
26  #endif
```

Q2 Application File

Application file (app.cpp) to test the Time class

```
1  /*************************************************************
2   * The application file app.cc to use the Time class         *
3   *************************************************************/
4  #include "time.h"
5
6  int main ()
7  {
8       // Instantiation of a time object
9       Time time (4, 5, 27);
10      // Printing the original time
11      cout << "Original time: " ;
12      time.print();
13      // adding 143500 seconds to the original time
14      for (int i = 0; i < 143500; i++)
15      {
16          time.tick ();
17      }
18      // Printing the time after 143500 ticks
19      cout << "Time after 143500 ticks " ;
20      time.print();
21      return 0;
22 }
```

KU KONKUK UNIVERSITY

Homework 1 Question 3

Q3 An Array to Represent the Number of Days in a Year

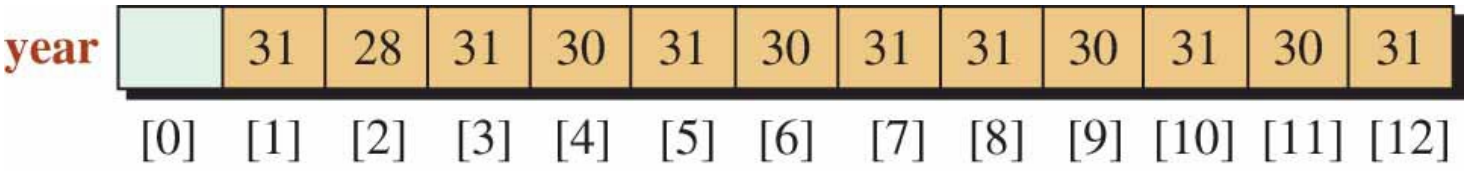
Although most of the time we use zero-indexing in the array, sometime, it is convenient to create an array of one extra element, and ignore the element at index 0.

In this case, our indexing starts from 1.

For example, we can create an array of 12 elements to hold the number of days in each month (for a non-leap year).

However, it is convenient to create an array of 13 elements and do not use the element at index 0.

An array to represent the number of days in a year



Question 3

- Create an array of 13 elements to hold the number of days in each month. Implement the code to get the same result with the following outputs.

```
Run:
Enter the month number (1 to 12): 1
There are 31 days in this month.
```

```
Run:
Enter the month number (1 to 12): 6
There are 30 days in this month.
```

```
Run:
Enter the month number (1 to 12): 0
Enter the month number (1 to 12): 13
Enter the month number (1 to 12): 3
There are 31 days in this month.
```

KU KONKUK UNIVERSITY

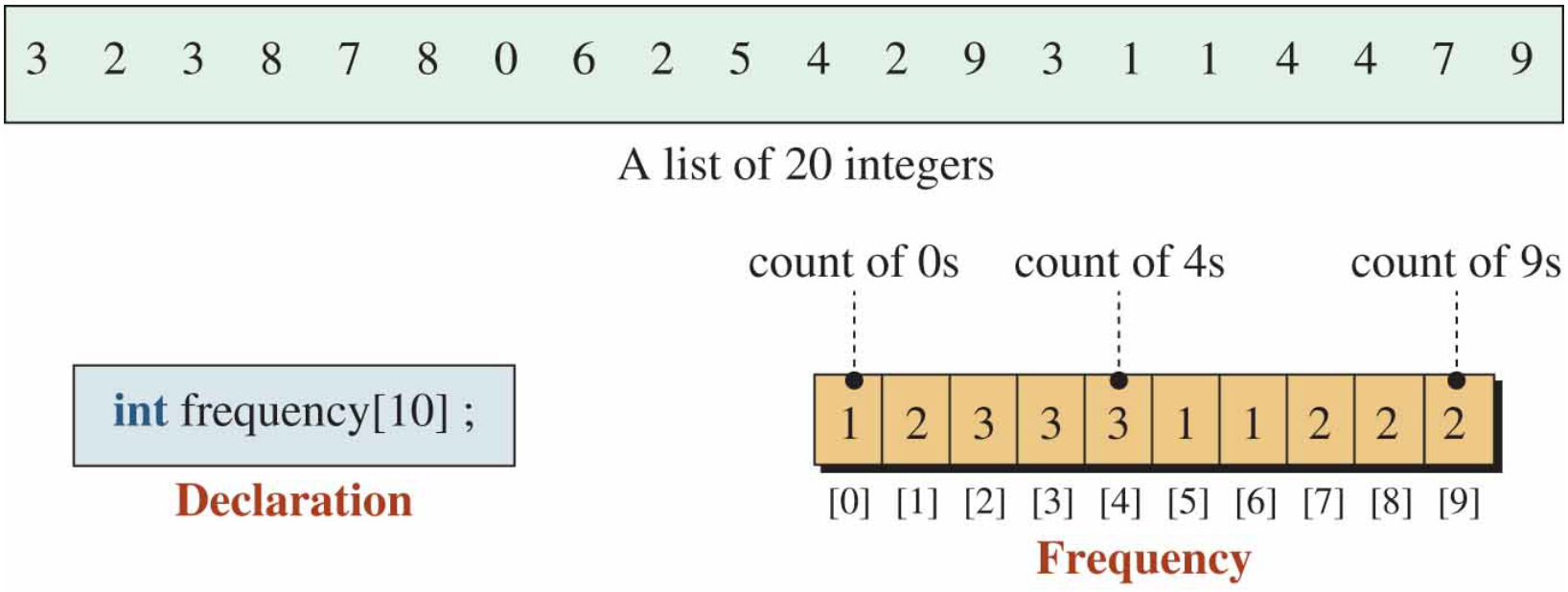
Homework 1 Question 4

Q4 Frequency Array and Histogram

One of the applications of arrays is to create a frequency array that shows the distribution of elements in a list of integers.

The array can then be used to create a histogram, a graphical representation of the table.

List of integers and frequency array



Q4 Frequency Array and Histogram

Understand Problem

We need to create an array frequency, in which each integer read from the list is related to the index of the array.

Develop Algorithm

We definitely need to go through four steps as shown below:

1. **We need to declare and initialize the frequency array to all 0's.**
2. **We need to open the integer file and make sure that the file is opened correctly; otherwise, the program should be terminated**
3. **We need to read the integer file, one integer at a time. If the integer is in the desired range (0 to 9 in our case), we increment the content of the frequency array at the corresponding index. If it is not, we ignore it.**
4. **We need to print the values stored in the histogram array, element by element. At the same time we need to create the corresponding line of the histogram.**

Q4 Frequency Array and Histogram

- Create a file using the given the integer list. Write a program to show the same result with the following output.

```
Run:
  There are 202 valid data items.
  0 ********************** 22
  1 **** 4
  2 ******************* 19
  3 ************ 13
  4 ******************** 20
  5 ********************************** 33
  6 **************************** 28
  7 **************************** 28
  8 ************************* 25
  9 ********** 10
```

The following shows the contents of the integer file for this problem.

```
1 3 2 2 5 7 3 2 8 0 6 4 6 7 0 7 8 5 4 2 3 0 6 7 5 8 5 4 8 9 6 5 5 9 2 3 5 2 6 7 8
0 6 4 6 7 0 7 8 5 4 2 13 0 6 7 5 8 5 4 8 9 6 7 0 7 8 5 4 2 3 0 6 7 5 8 5 4 8 9 6 5
2 7 0 7 8 5 4 2 3 0 6 7 5 8 5 4 8 9 6 15 5 9 7 0 7 8 5 4 2 3 0 6 7 5 8 9 6 5 6 6 4
5 9 7 0 7 8 5 4 2 3 0 6 7 5 8 9 6 5 6 6 4 2 7 0 7 8 5 4 2 3 0 6 7 5 8 5 4 8 9 6 5
1 3 2 2 5 7 3 2 8 0 6 4 6 17 0 7 8 5 4 2 3 0 0 6 4 6 7 0 7 8 5 4 2 3 0 6 7 5 8 1 1
```

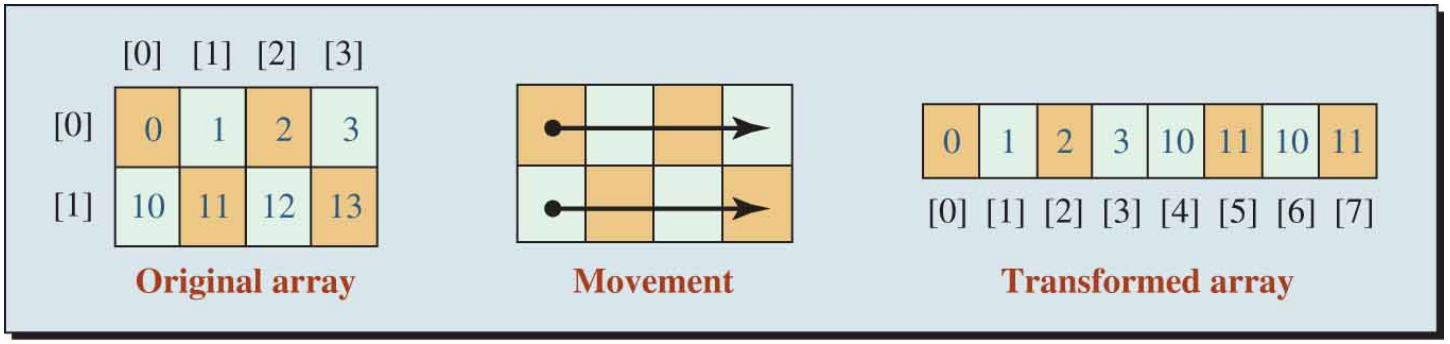
KU KONKUK UNIVERSITY

Homework 1 Question 5

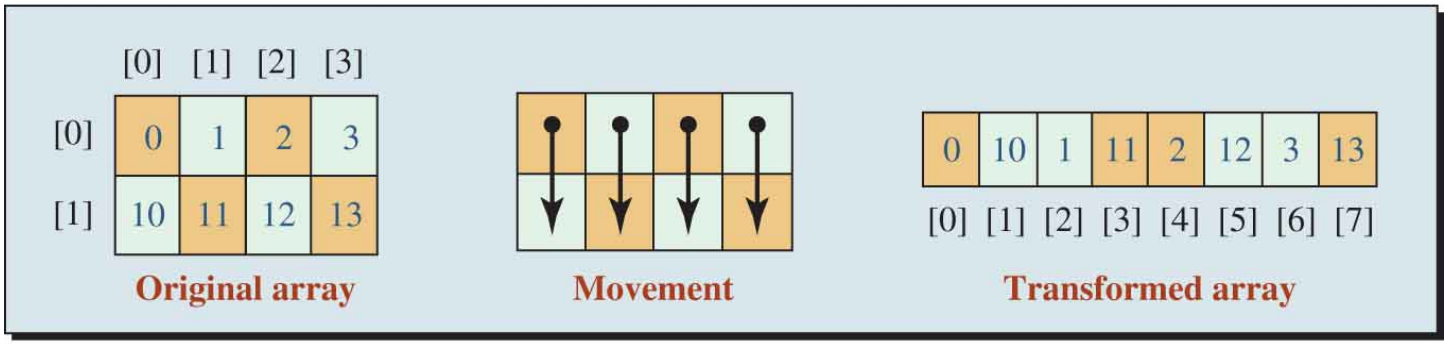
Q5 Linear Transformation

In linear transformation of a two-dimensional array to a one dimensional array, we have two choices: row transformation and column transformation

Linear Transformation



Row-by-row transformation



Column-by-column transformation

Q5 Linear Transformation

Understand Problem

We need to come up with the logic to change the data in an array of N rows and M columns of elements to a line with N by M elements.

Develop Algorithm

We declare and initialize a two-dimensional array.

1. **The first function transforms the array row by row and creates an array we call the rowArray.**
2. **The second function transforms the array column-by-column and creates an array we call the colArray.**
3. **The third function prints the two-dimensional array.**
4. **The fourth function prints either of the one-dimensional arrays.**

Q5 Linear Transformation

- Given the predefined two-dimensional array, write a program to show the same output below:

```
Run:
    Original Array
    0   1     2    3
    10 11    12   13
Row-Transformed Array:    0  1 2  3 10 11 12 13
Column-Transformed Array: 0 10 1 11  2 12  3 13
```

- Implement the following functions in the program and call them in main function:
  - ✓ void rowTransform(…)
  - ✓ void colTransform(…)
  - ✓ void printTwoDimensional(…)
  - ✓ void printOneDimensional(…)

KU KONKUK UNIVERSITY

Homework 1 Question 6

Question 6

- Implement the *reverse* function to show the same output.

```
1  /*************************************************************
2   * The program shows how a function can reverse the elements   *
3   * of an array using a pointer.                                *
4   *************************************************************/
5  #include <iostream>
6  using namespace std;
7
8  void reverse (int* , int );
9
10 int main ()
11 {
12       // Array declaration and initialization
13       int arr [5] = {10, 11, 12, 13, 14};
14       // Calling function
15       reverse (arr, 5);
16       // Printing array after reversed
17       cout << "Reversed array: ";
18       for (int i = 0; i < 5; i++)
19       {
20           cout << *(arr + i) << " ";
21       }
22       return 0;
23 }
```

```
Run:
Reversed array: 14 13 12 11 10
```

KU KONKUK UNIVERSITY

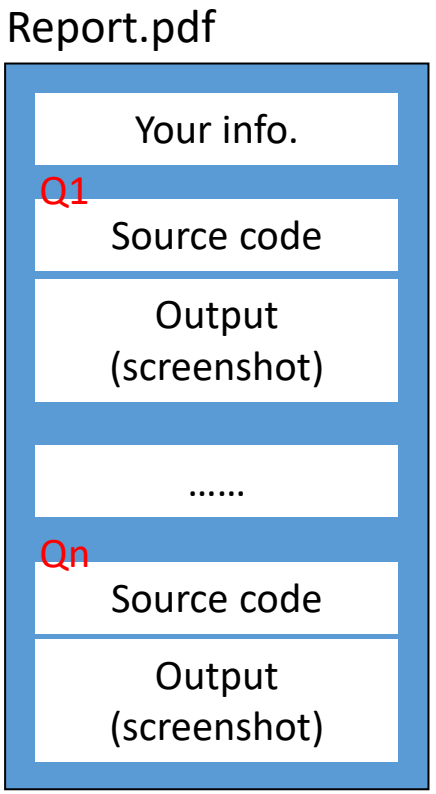
Submission

Report Submission

- Submission Guidelines:
  - ✓ Write your name, student ID, and major on the first page of the PDF file.
  - ✓ Attach your answer (code and output) for each questions in order.
    - ✓ Please ensure that you include your code in text format. Screenshots of your code will not be accepted for grading.
    - ✓ Attach a screenshot of each output.

- Submit a single PDF file to E-Campus

- Submission is due:
  - ✓ By April 2nd (Sub) 23:59
  - ✓ Late submissions will not be accepted.



End of Class

Thank you

E-mail: youngcha@konkuk.ac.kr