Lab 08  Classes

This Lab sheet involves exercises in 2 parts:

Guided Exercises (Q1) ➔ Skill-Drilling Exercise (Q2) ➔ Take-Home Exercise (Q3)

Q3 is IMPORTANT
(Please finish before Lecture09 and Lab09!!)

Reminder: Do Assignment 1

Guided Exercises

Q1. The Date Class (Version 3)

Refer to question 2 in Lec08_Ex.pdf (page 4) – available at the course web.

Download all files for this question. Hints are given in Date.h and Date.cpp.

Complete the "blanks" only

Finish the program and test in PASS.

Checkpoints (Make sure you fulfill the items below. Put a ✓ where appropriate. Answer questions marked with *.)

☐ I know how to write constructors, I know how they are called and how they run.
  - they run when main declares the objects d and __________.*, and when the next*/ isEndOfMonth* function creates the resultant object!!
    { choose one of them * }

☐ I know how to write member functions,
  ☐ I know how they are called, eg. by the main function
  ☐ I know they run.

☐ I notice that return type of the next*/ isEndOfMonth* member function is DATE!!
    { choose one of them * }

☐ I notice how the next*/ isEndOfMonth*/ valid* member function calls the next*/ isEndOfMonth*/ valid* member function, and next*/ isEndOfMonth*/ valid* calls the next*/ isEndOfMonth*/ valid* member function.

☐ I notice how calling objects are processed.
  The calling object ( Date d; in the main function) is * / is not * accessed by next
  The calling object ( Date d; in the main function) is * / is not * accessed by isEndOfMonth
  The calling object ( Date d; in the main function) is * / is not * accessed by valid.
  [All of the above should be true!!]

Any doubt? Ask helper!!

** Show this part to the Lab Helper and get the handout for: Skill-Drilling Exercises
** But if the helper is busy, simply go to the front-desk and self-serve (Model answer is available there).

Helper’s signature: _______
Question 1

/* Date.h */

class Date
{
public:
    // [YOUR TASK] for Lab08-Q1:
    // PLEASE ADD 2 lines below for the constructors.
    // (Recall: a constructor runs when we have "Date d;", "Date d(2009,1,1);" etc.)
    // (What to do here? Refer to coins.h - bottom of Lecture Notes Page 10.)

    __________________________
    __________________________

    // [GIVEN] for Lab08-Q1 - The function below checks whether the calling date object is the ending day of any month
    bool isEndOfAMonth();

    // [GIVEN] for Lab08-Q1 - The function below returns the "tomorrow" of the calling date object
    Date next();

    // I-O
    void input();
    void output();

    // Access functions
    void set(int new_y, int new_m, int new_d);
    int get_year();
    int get_month();
    int get_day();

private:
    // check if y, m, d are valid
    bool valid(int y, int m, int d);

    // check if y is a leap year
    bool isLeapYear(int y);

    // Date data
    int year;
    int month;
    int day;
};

/* Date.cpp */

#include <iostream>
#include "Date.h"
using namespace std;

// [YOUR TASK] for Lab08-Q1:
// PLEASE ADD 2 constructors.
// (Recall: a constructor runs when we have "Date d;", "Date d(2009,1,1);" etc.)
// (What to do here? Refer to coins.h - Lecture Notes Page 10.)

// Check whether the calling date object contains a date that’s the ending day of any month
bool Date::isEndOfAMonth()
{
    // [YOUR TASK] for Lab08-Q1:
    // *PLEASE COMPLETE this function:
    // if year, month and day+1 is a valid date <= use the valid member function!!
    //    then return false
    //    otherwise
    //    return true!!
    */
}

// Return the "tomorrow" of the calling date object
Date Date::next()
{
    // [YOUR TASK] for Lab08-Q1:
    // *PLEASE COMPLETE this function:
    //  if ( isEndOfAMonth() )  <= is the end of a month
    //      if it is not December, then
    //          return Date(year, month+1, 1);  <= Construct and return a date object (1st day of next month)
    //      else
    //          return ..  <= 1/1 of next year
    //  else
    //      return ..  <= the next day in the same month
    */
}
**Q2. Vending Machine**

**Step 1.** Locate Demo.exe for this question. **Run it** as shown below.

**The opening:**

The machine has $100 and 2 cokes, awaiting…

($5 per can; only accepts $1, $2, $5 coins)

Then YOU come →.

You may "press" the buttons: r, l, g, q (see below).

- You type 'r' // want refund
  → says: You haven’t paid any, so refund $0
- You type 'i 2' // insert $2
  → says: Alright, $2 so far
- You type 'i 2' // insert $2
  → says: Alright, $4 so far
- You type 'i 2' // insert $2
  → says: Alright, $6 so far
- You type 'g' // want a coke
  → says: OK, give you , refund $1, see you next time.
- You type 'g' // want a coke
  → says: You only paid $0 so far, No way!
- You type 'i 5' // insert $5
  → says: Alright, $5 so far
- You type 'i 3' // insert $3 (a $3 coin??)
  → says: Sorry! I only want $1, $2, $5!! Result: still $5 so far
- You type 'g' // want a coke
  → says: OK, give you (no extra for refund), see you next time.
- You type 'i 5' // insert $5
  → says: Alright, $5 so far
- You type 'g' // want a coke
  → says: Sorry, though you've paid $5, but I have no more coke.
- You type 'q' // quit: turn off the machine (forgot to refund!!)
  → says: Alright, at closing, I have $115 and 0 coke.

Make sure you can run it like the above.

Put a ✓ here when finished ✗.

Any question? Ask helper.
Step 2. Read the main function:
(Given at the course web.)

Note:
(1) The VendMach class represents a vending machine.
   - This line will run the constructor with 2 parameters.
(2) We often use switch-case to handle user commands like these.
(3) The member functions - insertCoin, getCoke, refund, status:
   Note how they are called in response to user commands:
   
   - refund
   - i 1: insert $1 coin
   - i 2: insert $2 coin
   - i 5: insert $5 coin
   - g: get a coke
   - q: quit

Data to handle (3 data members only):

We keep the number of Cokes, the amount of money owned by the machine,
and also the amount of money paid by a customer so far.
(For simplicity, we assume there are always enough coins to refund the customer.)

Examine the work of constructor, insertCoin, getCoke, refund, status (refer to the run-down in last page),
and figure out how they use and update the 3 data members.
Put a ✓ here when finished.

[CODING]
Step 3. [Refer to the hardcopy of the source code]
Firstly fill in the "blanks" in VendMach.h according to the Guidelines inside.
Then fill in the "blanks" in VendMach.cpp according to the Guidelines inside.

Test your program and then submit to PASS.

Checkpoints (Make sure you fulfill the following items and put a ✓ where appropriate.)

☐ I have successfully finished the program in this part (100% correct in PASS).
☐ I understand how everything in the program works.

** Show this part to the Lab Helper and get the handout for: Take-Home Exercises
** But if the helper is busy, simply get go to the front-desk and get the handout for: Take-Home Exercises 😊
Question 2

/*VendMach.h*/
#include <iostream>
#include "VendMach.h"
using namespace std;

/*[Lab08] Given framework for students*/

// The Vending Machine class
class VendMach{
public:
    // [YOUR TASK]
    // List the prototypes for the constructor and 4 public member functions
    // (First 2 have been done for you.)
    VendMach(int tm, int tc); // Given: Constructor (In the given main function, it runs at "VendMach m(100,2);")
    void insertCoin(int x); // Given: insertCoin (Read how it is called by the main function !!)
    ___________________   // Todo: add the getCoke, refund, status member functions
    ___________________   // Match how they are called by the main function !!
    ___________________

private:
    // [YOUR TASK]
    // Add the 3 data members (totalMoney, totCokes, paid) mentioned in the lab sheet.
    ___________________

};

//VendMach.cpp/
#include <iostream>
#include "VendMach.h"
using namespace std;

/* [YOUR TASK]

FIRSTLY finish VendMach.h,

AFTERWARDS, complete all functions below.
*/

// (1) Constructor - simply set 3 data members (In the given main function, it is called at "VendMach m(100,2);"
VendMach::VendMach(int tm, int tc)
{
    paid=0;
    __________=tm;
    __________=_________
}

// (2) The member function - status: Show current status of the machine
void VendMach::status()
{
    cout << "$" << totalMoney+paid << " and " << totCokes << " coke(s)\n";
}

// (3) The member function - refund: Refund to user
    ___________________
    { // Refund the money inserted by the user so far (based on value of paid), ie.
        // output (cout) like "$1 refunded" and reset the value of paid to zero.
        // (2 lines only!!)
    }

// (4) The member function - insertCoin: Accept user's coin x (if valid)
    ___________________
    { // If x is valid ($1, $2, $5), then update the value of paid.
        // No matter x is accepted or not, display (cout) the amount paid so far, like "$6 paid".
        // (3 lines only!!)
    }

// (5) The member function - getCoke: Give the user a coke/Refund to user (if valid)
    ___________________
    { /* If enough money and still has stock,
        update the 3 data members and say "Here is your Coke\n"
        (if excess was paid, refund it: "Here is your Coke, $? refunded\n". <= see/call the refund function !)
        Otherwise output "Not enough money\n" or "Out of stock\n" accordingly
        */
    }
Take-Home Exercises

Q3 is IMPORTANT (Please finish before Lecture09 and Lab09!!)

Before proceeding, check (√) the points below:

☒ I’m okay with the lecture notes, examples, and lecture exercises covered during all previous lectures.
☒ I’m okay with all previous lab exercises. (Or at least I’ve read the answers and I’ll try to find time to redo them)
☒ If I have any problem with these exercises, I may contact Helena or go to Programming Clinic (see course web).

Q3. Fraction - Version 1 (Member Functions accessing the calling object plus another one)

(A) Preparation: Understanding Fraction operations

Your task: According to the answer for (a) below, finish (b).

(a.) What is the sum of \( \frac{5}{6} + \frac{2}{7} \)?

Working:

\[
= \frac{5 \times 7 + 2 \times 6}{6 \times 7} \\
= \frac{47}{42} \quad \text{(Find the HCF of 47 and 42: obtain 1)} \\
= \frac{47+1}{42+1} \\
= \frac{47}{42} \quad \text{(The answer) ← No need to turn it to “1 and 5/42”}
\]

(b.) What is the sum of \( \frac{5}{6} + \frac{2}{3} \)?

Working:

\[
= \\
= \\
= \quad \text{(Find the HCF of and : obtain )} \\
= \\
= \quad \text{(The answer)}
\]

(B) We will write a Fraction class which keeps 2 data members (numer, denom), like:

\[
\begin{align*}
1 & \leq \text{nominator (numer)} \\
2 & \leq \text{denominator (denom)}
\end{align*}
\]

where numer and denom are relatively prime (i.e., numer and denom have no common factors).

Your task: Read the code for the Fraction class below. Examples are given on next page.

DONT HURRY TO START CODING - MUST FOLLOW THE GUIDELINE IN P.5-6

```cpp
class Fraction {
public:
    Fraction();  // Initialize the fraction with numer=0, denom=1
    Fraction(int n, int d);  // Initialize the fraction with numer=n, denom=d
    void display();  // Show (cout) the fraction like 1/2
    void add(Fraction num);  // +, -, x, ÷ (See Examples on next page)
    void sub(Fraction num);
    void mul(Fraction num);
    void div(Fraction num);
private:
    void reduce();  // Simplify the fraction. Eg. if it is 27/18 (ie. numer=27 and denom=18),
                    // then change it to 3/2 (ie. numer becomes 3 and denom becomes 2).
    int numer, denom;
};
```
Example 1

Fraction num1(1,6), num2(1,2);
num1.display(); // display (cout) 1/6
num2.display(); // display (cout) 1/2
num1.add(num2); // add 1/2 to num1 => num1 becomes 2/3, but num2 doesn't change
num1.display(); // display (cout) 2/3

Example 2

Fraction num1(5,6), num2(2,3);
num1.add(num2); // add 2/3 to num1 => num1 will be 5/6+2/3, i.e. 27/18, then simplified to 3/2; but num2 doesn't change
num1.display(); // display (cout) 3/2 (No need to turn it to "1 and 1/2")

(C) Your task: Create Fraction.h and Fraction.cpp according to the class definition in last page.

Please follow the Guidelines on next page.
[This time you are allowed to copy from Lab08_page6.pdf. But remember to fix the indentations first!!]

The driver program (main) is given at the course web (also shown here). (A driver program is a simple program for testing your code.)

```cpp
#include "fraction.h"
#include <iostream>
using namespace std;

void main()
{
    int numerator, denominator;
    Fraction num1, num2, ans;
    char dummy;
    cout << "Input a fraction (a/b): ";
    cin >> numerator >> dummy >> denominator;
    num1 = Fraction(numerator, denominator);
    cout << "Input another fraction (a/b): ";
    cin >> numerator >> dummy >> denominator;
    num2 = Fraction(numerator, denominator);
    // addition
    ans = num1;
    ans.add(num2);
    num1.display();
    cout << " + ";
    num2.display();
    cout << " = ";
    ans.display();
    cout << endl;
    // .. sub, mul, div ...
    ...
}
```

Sample input and output:
(Underlined contents are input by the user.)

Input a fraction (a/b): 1/2
Input another fraction (a/b): 2/3
1/2 + 2/3 = -1/6
1/2 * 2/3 = 1/3
1/2 / 2/3 = 3/4
Press any key to continue . . .

Special cases (To be handled by .display())

Case 1: Denom is 1, then show numer only.
    eg., Suppose num1 contains 8/1.
    Then num1.display() shows 8 only.

Case 2: Numer is 0, then show 0 only.
    eg., Suppose num1 contains 0/1.
    Then num1.display() shows 0 only.

Remember to call reduce() at the end of each operation to simplify the fraction.
(Reminder: in the constructor with 2 parameters, also call reduce().)

Assume that no error case will happen (eg. no division-by-zero).

Submit Fraction.h and Fraction.cpp to PASS (DO NOT USE OTHER FILE NAMES).
** NO NEED to upload the driver program (main function) to PASS.
Guidelines

Fraction - Version 1 (Member Functions which access the calling object plus another one)

/* fraction.cpp */
#include <iostream>
using namespace std;
#include "fraction.h"

//Default constructor
Fraction::Fraction()
{ ..// set values for numer and denom as 0 and 1 }

//Constructor
Fraction::Fraction(int n, int d)
{ ..// set values for numer and denom as n and d
  reduce(); // "n/d" may not be in simplest form. So reduce it. }

//Display a fraction
void Fraction::display()
{ ..// eg. showing "1 / 2" }

/* Find the highest common factor between 2 positive integers. E.g., say x1=12 and x2=8, their HCF is 4 */
int findHCF(int x1, int x2)
{ ..// see Lec08_exercise Page 1 }

//Reduce the Fraction
void Fraction::reduce()
{ int factor=findHCF(abs(numer),abs(denom)); // call findHCF to get the factor. 
  numer=numer/factor;
  denom=denom/factor;
}

//Add a fraction
void Fraction::add(Fraction num)
{ int new_denom, new_numer;
  new_numer=numer*num.denom+num.numer*denom;
  new_denom=denom*num.denom;
  numer=new_numer;
  denom=new_denom;
  reduce();
}

/* fraction.h */

//The Fraction Class
class Fraction {
public:
  //Constructors
  Fraction();
  Fraction(int n, int d);
  void display();
  void add(Fraction num);..
private:
  void reduce();
  int numer, denom;
};