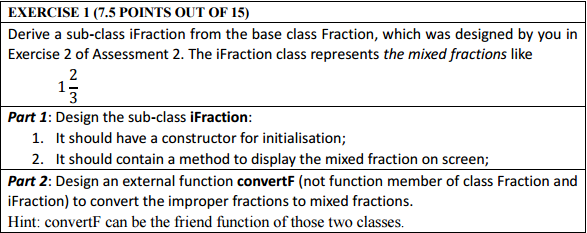
**Assignment 4**

**1301058**

**Zhang Junming**

**Exercise 1**

Question



Model Answer

Software Development Process

1. **Problem statement**

In part 1: Design a sub-class iFraction of class Fraction in Assignment 2 which could represents the mixed fraction, at the same time, write the constrictor and display function for iFraction.

In part 2: Write an external function to convert improper fractions to mixed fractions.

1. **Analysis**

Inputs:

1. Fraction: Construct part (two numbers, one represents numerator, another represents denominator)
2. iFraction: Construct part(one number to represent integer part)
3. In part2: Convert function, ask user to create an improper fraction

Outputs:

1. Display the improper fraction
2. Display the mixed fraction
3. Display the improper fraction which user input and then display the mixed fraction which convert from the improper fraction

Additional requirements or constraint

The convert function should be an external function, it means that convert function is not the function member of class

1. **Design**

Fraction.h

1. Add header file Fraction.h

<1> use #ifndef and #define to prevent repetitive include

<2> include library “iostream”

<3> using of the ste namespace

<4> define a class which called Fraction

Private

int top – represent the numerator

int bottom – represent denominator

Public

Fracion() – default constructor

<1> let top equal to 0 and bottom equal to 1

Fraction(int N,int D) – normal constructor

<1> let top equal to N and bottom equal to D

void out () – display the fraction on screen

<1> display numerator first, then display “/” and the denominator

<5> add class iFraction and iFraction convertF to friend

<6> use #endif

1. Add header file iFraction.h

<1> use #ifndef and #define to prevent repetitive include

<2> include library “iostream” and “Fraction.h”

<3> using of the ste namespace

<4> define a private type sub-class which called Fraction and it inherit from class Fraction

Private

int number – represent the integer part

Public

iFracion() – default constructor

<1> let number equal to 1

iFraction(int Number,Fraction Term) – normal constructor

<1> let number equal to Number, top equal to top in Term and bottom equal to bottom in Term

void display () – display the fraction on screen

<1> display number first, then display the fraction part

<5> add class Fraction and iFraction convertF to friend

<6> use #endif

1. Write external function int max(int x,int y) to find the greatest common divisor in fraction

<1> int i – to store the remainder

<2> there are four conditions:

i numerator and denominator are positive

ii numerator is negative and denominator is positive

iii numerator is positive and denominator is negative

iiii numerator and denominator are negative

setting up a loop let i equal to numerator / denominator and then let numerator equal to denominator and let denominator equal to i until i equal to 0, at this time, the value of numerator is the greatest common divisor

for i and iiii, return x

for ii and iii, return -x

1. Write convert function convert: iFraction convert (Fraction term)

<1> int k – represent the greatest common divisor

<2> int i – represent the remainder of fraction

<3> iFraction result – to store mixed fraction

<4> using external function max to obtain the greatest common divisor

<5> numerator and denominator divide their greatest common divisor at the same time

<6> let i equal to numerator divide denominator

<7> there are four conditions

i numerator and denominator are positive

ii numerator is negative and denominator is positive

iii numerator is positive and denominator is negative

iiii numerator and denominator are negative

for i and iiii

if their remainder small than 1

keep the original numerator and denominator

but if the numerator and denominator are negative, let them become positive and let integer part \* -1

give the related value to result

if their remainder big than 1

let integer part equal to reminder

let numerator equal to original numerator – original denominator\*i

if the numerator and denominator are negative, let them become positive and let integer part \* -1

give the related value to result

for i and iii

it similar as i and iiii but change the negative value to positive first and adjust the positive and negative after convert

1. Write the main function

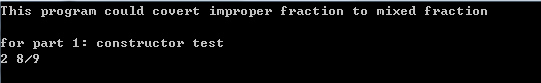
<1> tell user the function of this program

<2> ask user to input an improper fraction

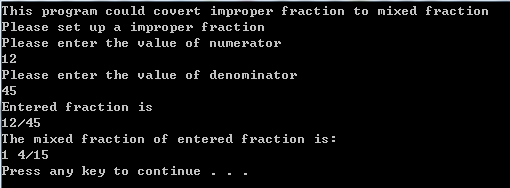
<3> convert it to mixed fraction

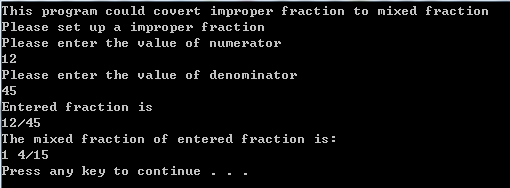
<4> display the improper fraction and mixed fraction on screen

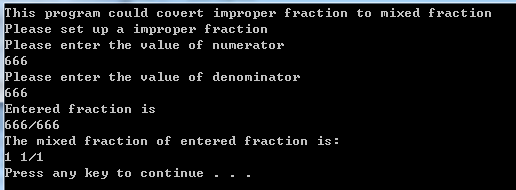
1. **test**:



Part 1: constructor



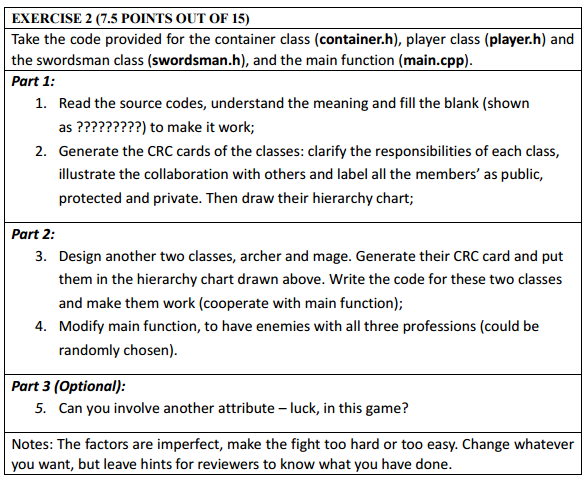




Part 2: test

**Exercise 2**

Question



Model Answer

Software Development Process

**1**. **Problem statement**

In part 1: Add codes to complete a game, totally 7 blanks. Let this game run successful and then generate a CRC cards to explain the relationship between each class

In part 2: Add two new professions to this game and let this game run successful, add them to the CRC in the end

**2. Analysis**

Inputs:

Orders to control your role to fight with one enemy

Outputs:

The order result and weather win this game

**3. Design**

Blank 1:

#ifndef \_CONTAINER :

#ifndef means if not define , it works with #define, they could prevent repeated include to program

Blank 2:

numOfHeal--;

numOfHeal means the number of Heal Water, it could add blood to character, bool container::useHeal() this function means use Heal Water, therefore, the number of Heal Water should minus 1 when use this function

Blank 3:

bag.set(bag.nOfHeal()+p.bag.nOfHeal(),bag.nOfMW()+p.bag.nOfMW());

it means get the Heal and Magic water from died enemy, this function is:

void player::transfer(player &p)and this function will run when enemy died, the number of people’s Heal and Magic water will plus enemy’s

Blank 4:

void showinfo(player &p1, player &p2)

This function name found in player.h header files and is a friend function to class player

Blank 5:

class swordsman:public player

Base on the note after this code: subclass swordsman publicly inherited from base player it could infer that class swordsman is a sub-class of class player and the type of inherit is public

Blank 6:

Delete human;

Note is: player is dead, program is getting to its end, what should we do here?

It could infer player is died, therefore delete the data of player

Blank 7:

Nothing

Note is: You win, program is getting to its end, what should we do here?

When player win all enemy, the following code will display congratulations, therefore I think there does not need other code, maybe it could create a file to store the winner’s name.

Another two classes:

Archer:

Similar to swordsman, create a header file which called archer and a cpp file which called archer

<1> change the enumerate type of job to ar

<2> reduce the original HP and the development of it

promote the original AD to improve attack ability

promote the speed to improve the rate of evade

archer should be a high attack and low HP people, therefore I change the normal attack calculate formula to improve archer’s attack ability

Magical man:

Similar to swordsman, create a header file which called migicman and a cpp file which called migicman

<1> change the enumerate type of job to mg

<2> promote the original MP and the development of it

promote the original AD to improve special attack ability

reduce the speed to decrease the rate of evade

magic man should has a high special attack and low HP, therefore I change the special attack calculate formula to improve mage’s spcial attack ability

Enemies randomly choose their professions

<1> add “time.h” library

<2> set the seeds of time for generate different random number

<3> generate random number between 1~3 and store in int choose

<4> judge the value of choose, if choose equal to 1, set enemy to swordsman; if choose equal to 2, set enemy to archer; if choose equal to 3, set enemy to magic man

**CRC cards:**

|  |
| --- |
| CLASS CONTAINER |
| Protected:   1. int numOfHeal to represents the number of Heal Water 2. int numOfMW to represents the number of Magic Water |
| Public:  container(): Default constructor  void set(int heal\_n,int mw\_n); give player Heal and Magic water  int nOfHeal() obtain the number of Heal Water  int nOfMW() obtain the number of Magic water  void display() display the related information of player  bool useHeal() judge weather use the Heal Water  bool useMW() judge weather use the Magic Water |

|  |
| --- |
| CLASS PLAYER |
| Let function showinfo and classed swordsman, archer and magicman be the friend to class player |
| Protected:  declare the property of player such like HP,HPmax…..  string name to store the name of player  job role – represents the three different job  declare bag which is a class of container to store the Heal and Magic water of player |
| bool attack: judge weather use normal attack.  bool specialatt: judge weather use special attack.  isLevelUp: judge weather level up  reFill(): reset the HP and MP when start a new fight  bool death: judge weather player dead  isDead(): check whether player dead  bool useHeal(): judge weather use Heal Water  bool useMW(): judge weather use Magic Water  transfer(): let the Heal and Magic water to player when win the enemy  shwoRole(): display the job of player |
| Private:  bool playerdeath: judge weather user death and weather need to inherit |

|  |
| --- |
| CLASS OF THREE JOB |
| Public:  archer(): normal constructor  isLevelUp(): player is level up  bool attack: weather use normal attack  bool specialatt: weather use special attack  AI(): control the action of enemy, use which type attack, weather use Heal or Magic water |

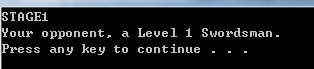
**Hierarchy Chart**

****

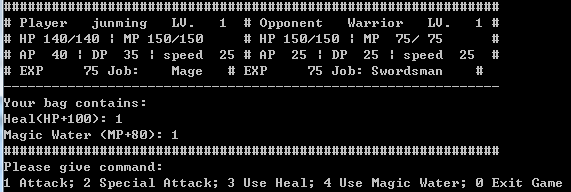
1. **Test**:



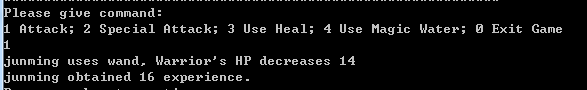
Create player



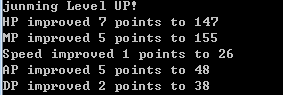
Enemy



Interface



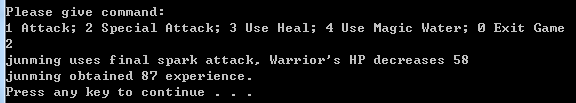
Use normal attack



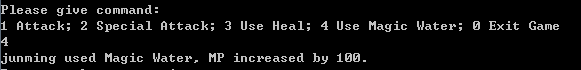
Level up



Enemy action



Use special attack



Use magic water



Enemy died



Win



When player died



Random job of enemy