University of Toronto
Faculty of Applied Science and Engineering

ECE 244F

PROGRAMMING FUNDAMENTALS

Fall 2019

Midterm Test

Examiners: T.S. Abdelrahman and M. Shaghaghi

Duration: 110 minutes

This test is OPEN Textbook and CLOSED notes. The use of computing and/or communicating devices is NOT permitted.

Do not remove any sheets from this test book. Answer all questions in the space provided. No additional sheets are permitted.

Work independently. The value of each question is indicated. The total value of all questions is 100.

Write your name and student number in the space below. Do the same on the top of each sheet of this exam book.

Name: _____
(Underline last name)

Student Number: _____

Q1. _____	**Q8.** _____
Q2. _____	**Q9.** _____
Q3. _____	**Q10.** _____
Q4. _____	**Q11.** _____
Q5. _____	**Q12.** _____
Q6. _____	**Q13.** _____
Q7. _____	**Q14.** _____

Total

Important Notes

1. In answering the questions, you must assume the C++ 11 standard and the use of the `g++` compiler available on the Linux machines in the ECF labs.

2. There are some multiple-choice questions for which incorrect answers carry part marks. This is indicated in the relevant questions.

3. If your have any doubts about a question, write your assumption down. If they are sensible, they will be taken into account.

Question 1. (8 marks). *Warm Up Questions.*

Answer the following questions by circling the most appropriate answer.

(a) The default access control specifier for class members is _____.

a. private
b. public
c. both private and public
d. Neither private nor public

(b) A user defined header file called file.h is included in a program as _____.

a. #include <file>
b. #include "file.h"
c. #include "file"
d. #include file.h

(c) `cin` is _____.

a. a class
b. an object
c. a package
d. a namespace

(d) The operator to access members of an object using the objects name is _____.

a. .
b. ->
c. *
d. None of the above

(e) Friend functions of a class are _____.

a. functions that are called "friend"
b. functions allowed to access private members of the class
c. functions allowed to access public members of the class
d. b. and c.
e. All of the above

(f) The `this` keyword in a method gives _____.

a. the object on which a method is invoked
b. a pointer to the method that is invoked
c. a pointer to the object on which the method is invoked
d. a pointer to the class to which the method belongs

(g) The copy constructor is executed when _____.

a. an object is assigned from another object of the same type
b. an object is created as a copy of another object of the same type
c. an object is passed to a function and the pass mechanism is by reference
d. b. and c.
e. None of the above

(h) One key advantage of separate compilation is _____.

a. to use more command to compile and link
b. to speed up the program development cycle
c. to discover more bugs
d. There is really no advantage

Question 2. (8 marks). *Classes.*

Consider the definition of a class called `Nova`, which is in the file `Nova.h`.

```
#include <iostream>
using namespace std;

class Nova {
  private:
    // Private members not shown

  public:
    // Public members not shown

};
```

Now consider the following program that uses the `Nova` class. The program compiles and runs correctly.

```
#include <iostream>
using namespace std;

#include "Nova.h"

int main () {
  Nova a(3,8.1);
  Nova* p;
  Nova b(a);
  ++a.it;
  a.setAll(1,7.8);
  p = new Nova(9,12.7);
  if (a != b) *p = a + b;
  delete p;

  return 0;
}
```

What members of the class `Nova` **must** exist for the above code to compile with no errors? Give variable declarations and/or method prototypes in the table below. Note that you may or may not need to fill every row in the table.

Question 3. (8 marks). *Classes and Objects.*

Consider the following class definition and implementation. You may assume they are error free.

```
#include <iostream>
using namespace std;

class Fraction {
    int numerator;
    int denominator;
  public:
    Fraction (int x, int y);                    // Method 1
    Fraction (const Fraction& source);          // Method 2
    ~Fraction ();                               // Method 3
    Fraction& operator= (Fraction & rhs);       // Method 4
    void print(Fraction obj) const;             // Method 5
};

Fraction::Fraction(int x, int y) {
    numerator = x;
    denominator = y;
}

Fraction::Fraction(const Fraction& source) {
    numerator = source.numerator;
    denominator = source.denominator;
}

Fraction::~Fraction() { }

Fraction & Fraction::operator=(Fraction& rhs) {
    numerator = rhs.numerator;
    denominator = rhs.denominator;
    return (*this);
}

void Fraction::print(Fraction obj) const {
    cout << "(" << obj.numerator << "/" << obj.denominator
         << ")" << endl;
}
```

Assume each of the following snippets of code, is in a `main` function, where `iostream` is included and the `std namespace` is used. For each code snippet, indicate which methods are invoked.

Treat <u>each code snippet by itself</u>, independent of the other code snippets. Circle **only one** answer.

Note: while the correct answer gets full marks, <u>some incorrect answers get part marks</u> (and some get none).

1. `Fraction* undefined;`

    a. None
    b. Method 1
    c. Methods 1 and 2
    d. Methods 1 and 3
    e. Methods 1 and 4
    f. Methods 2, 3 and 4
    g. The code snippet results in a compile time error

2. ```
   Fraction threeQuarters (3,4);
   cout << threeQuarters.numerator << "/"
        << threeQuarters. denominator << endl;
   ```

    a. None
    b. Method 1
    c. Methods 1 and 2
    d. Methods 1 and 3
    e. Methods 1 and 4
    f. Methods 2, 3 and 4
    g. The code snippet results in a compile time error

3. `Fraction* undefined = new Fraction[100];`

    a. None
    b. Method 1
    c. Methods 1 and 2
    d. Methods 1 and 3
    e. Methods 1 and 4
    f. Methods 2, 3 and 4
    g. The code snippet results in a compile time error

4. ```
   Fraction threeQuarters (3,4);
   Fraction zeroPoint75 (threeQuarters);
   ```

    a. None
    b. Method 1
    c. Methods 1 and 2
    d. Methods 1 and 3
    e. Methods 1 and 4
    f. Methods 2, 3 and 4
    g. The code snippet results in a compile time error

5.
```
Fraction threeQuarters (3,4);
Fraction half (1,2);
half.print(threeQuarters);
```

a. None
b. Methods 1 and 5
c. Methods 1, 2, 3 and 5
d. Methods 1, 2 and 5
e. Methods 1, 4 and 5
f. Methods 2, 3, 4 and 5
g. The code snippet results in a compile time error

6.
```
Fraction threeQuarters (3,4);
Fraction sixeighth (6,8);
threeQuarters = sixeighth;
```

a. None
b. Method 1
c. Methods 1 and 2
d. Methods 1 and 4
e. Methods 1, 2 and 4
f. Methods 1, 2, 3 and 4
g. The code snippet results in a compile time error

Question 4. (8 marks). *Methods, Functions and Objects.*

Assume there exists three classes: `SuperHero`, `Villain` and `Winner`. These classes are implemented correctly and are available for use.

(a) Complete below the prototype of a non-member function called `Fight`. The function takes two arguments. The first is called `superman` that is an object of type `SuperHero`, passed by value. The second is an object of type `Villain` called `Luthor`, passed by value. The function returns by value an object of type `Winner`.

```
Fight (                                                    );
```

(b) Complete below the prototype of a non-member function called `CreateVillain`. The function takes two arguments. The first is called `batman` that is an object of type `SuperHero`, passed by value. The second is a pointer to an object of type `Villain` called `ptr`. The function allocates a new object of type `Villain` and stores its address in the second argument of the function, i.e., `ptr`. The function returns nothing.

```
CreateVillain (                                            );
```

(c) Complete below the prototype of a non-member function called `MakeSuperHero`. The function takes one argument called `kind` of type `string`, passed by value. It returns by reference an object of type `SuperHero`.

```
MakeSuperHero (                                            );
```

Question 5. (4 marks). *Functions and Objects.*

Consider a (non-member) function called `doIt`, which takes a single object of type `DayOfYear` and returns a single object also of type `DayOfYear`. You may assume that the class `DayOfYear` is correctly implemented and that `DayOfYear.h` is included. Which of the following implementations of this function is problem-free? Indicate your answer by placing an **X** in the appropriate column in the table.

Implementation	Problem-Free?	Has a problem?

`DayOfYear doIt(DayOfYear & arg) {`
`  DayOfYear temp;`
`  temp = arg;`
`  return (arg);`
`}`		

`DayOfYear doIt(DayOfYear & arg) {`
`  DayOfYear temp;`
`  temp = arg;`
`  return (temp);`
`}`		

`DayOfYear & doIt(DayOfYear & arg) {`
`  DayOfYear temp;`
`  temp = arg;`
`  return (*this);`
`}`		

`DayOfYear & doIt(DayOfYear & arg) {`
`  DayOfYear temp;`
`  temp = arg;`
`  return (temp);`
`}`		

Question 6. (4 marks). *C++ I/O.*

For each of the following main functions, indicate the output produced in response to the user entering `1 2 3 4 five` on the keyboard followed by the `Enter` key. Circle only one answer.

Note: while the correct answer gets full marks, some incorrect answers get part marks (and some get none).

(a) (2 marks).

```
#include <iostream>
using namespace std;

int main() {
    int num = 0;
    int  sum = 0;

    while (!cin.fail()) {
       cin >> num;
       sum = sum + num;
    }
    cout << sum << endl;
    return (0);
}
```

Circle one answer:

1. 6
2. 10
3. 14
4. None; the program runs in an infinite loop

(b) (2 marks).

```
#include <iostream>
using namespace std;

int main() {
    int num = 0;
    int  sum = 0;
    bool more = true;

    while (more) {
       cin >> num;
       if (cin.fail()) more = false;
       else sum = sum + num;
    }
    cout << sum << endl;
    return (0);
}
```

Circle one answer:

1. 6
2. 10
3. 14
4. None; the program runs in an infinite loop

Question 7. (9 marks). *C++ I/O.*

Write a C++ function `void readInts()` that repeatedly reads integers from the standard input (using `cin`) and then immediately outputs the input integer (using `cout`), one integer per line. When the end-of-file is reached, the function prints the message `End of File Reached` on a line by itself before returning. If a non-integer is input the function should print the message `Invalid Input` on a line by itself, should discard the rest of the stream and should continue reading integers again until the end-of-file is reached.

Write your code in the box below.

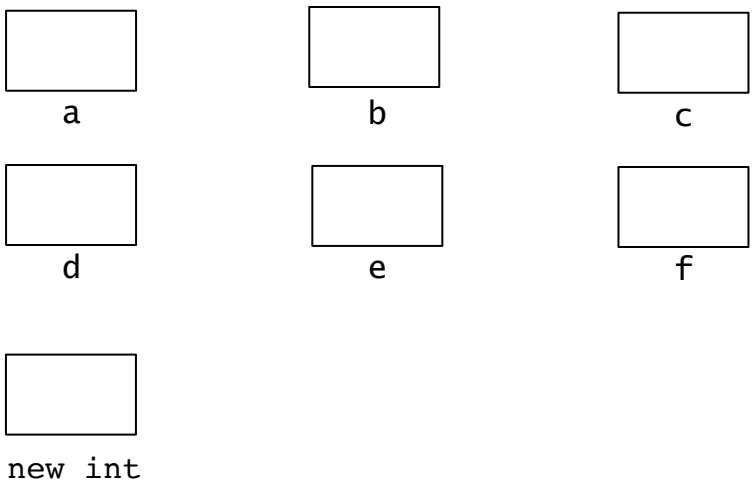
```
#include <iostream>
using namespace std;

void readInts() {









}
```

Question 8. (7 marks). *Pointers.*

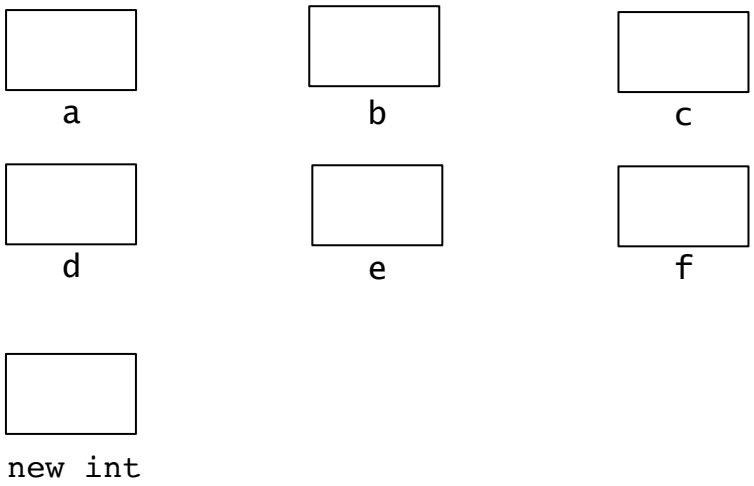
Assume that the following code compiles and runs properly.

```
int a = 6;
int* b = &a;

int* foo(int** c) {
    (**c)++;
    *c = b;
    int* d = new int;
    *d = 10;
    // Point #1
    return d;
 }

 int main() {
    int e = 7;
    int* f = &e;
    b = foo(&f);
    // Point #2
    return 0;
 }
```

(a) (3.5 marks). Complete the following diagram by showing the values of variables and/or pointers when execution reaches the point labeled "Point #1". For an integer variable, simply show the integer value inside the corresponding box. For a pointer, indicate the value of the pointer by drawing an arrow from the box corresponding to the pointer to the box corresponding to the variable the pointer points to.



(b) (3.5 marks). Complete the following diagram by showing the values of variables or pointers when execution reaches the point labeled "Point #2". For an integer variable, simply show the integer value inside the corresponding box. For a pointer, indicate the value of the pointer by drawing an arrow from the box corresponding to the pointer to the box corresponding to the variable the pointer points to. Cross out any variables (automatic, dynamic or pointers) that no longer exist.



Question 9. (5 marks). *Pointers and Arrays.*

In the blank space for each part **(a)** – **(d)** below, provide the declaration and initialization of the variable **`i`** (for example: `int i = 10;`). Your initialization of `i` must make use of the variable `j`, and lead to the expected output when used in the statements that follow it.

(a)
```
int j = 10;

// declare and initialize i:
```
[ ]
```
cout << *i << endl;
j = 5;
cout << *i << endl;
```

Expected Output of (a)

```
10
5
```

(b)
```
int j = 10;

// declare and initialize i:
```
[ ]
```
cout << i << endl;
j = 5;
cout << i << endl;
```

Expected Output of (b)

```
10
10
```

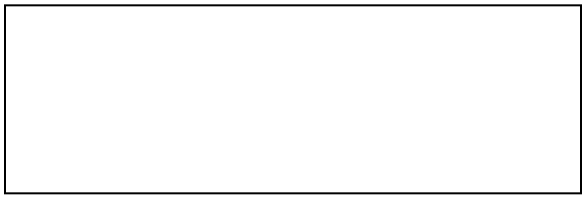
(c)
```
int j = 10;

// declare and initialize i:
```
[ ]
```
cout << i[0] << endl;
j = 5;
cout << i[1] << endl;
```

Expected Output of (c)

```
10
10
```

(d)

```
int j = 10;

// declare and initialize i:
```



```
cout << *(i[0]) << endl;
j = 5;
cout << *(i[1]) << endl;
```

Expected Output of (d)

```
10
5
```

Question 10. (10 marks). *Arrays and Objects.*

Consider the following modified and simplified definition/implementation of the class `Shape`, used in your lab assignment 3. You may assume the class is correctly defined/implemented.

```
#include <iostream>
using namespace std;
#include <string>

class Shape {
   private:
       string name;
       string type;
   public:
              Shape() { }
       string getName() const {return name;}
       string getType() const {return type;}
       void   setName(string n) {name = n;}
       void   setType(string t) {type = t;}
};
```

A `main` function dynamically allocates then de-allocates `n Shape` objects, along with other dynamically allocated variables, where `n` is an integer value read from `cin`. The code to de-allocate the objects and the other variables (so that no memory leak exists) is show below at the end of `main`.

Give the code to allocate the `n` objects and to set the type of each of the `n Shape` objects to the string `circle`. Assume `iostream` has been included and that the `std` namespace is used.

```
int main () {
   int n;
   cin >> n;

   // Write code to allocate objects and other variables here









   // Write code to set type of each Shape object to circle here




// De-allocate all dynamically allocated variables
   for (int i=0; i < n; ++i) {
       delete *(p[i]);
       delete p[i];
   }
   delete [] p;
   return (0);
}
```

Question 11. (5 marks). *Dynamic Memory Allocation.*

Study the following program and answer the questions below. You may assume that the class `Nova` is correctly defined an implemented.

```
#include "Nova.h"
void MostNova (Nova* y) {
   Nova* ptr = new Nova();
   delete [] y;
   Nova x;
   // Point Z
   return;
}

void MoreNova (Nova & x) {
    Nova a;
    // Point Y
    Nova b;
    Nova* ptr = new Nova[10];
    MostNova(ptr);
    // Point W
    return;
}

int main() {
    Nova a;
    Nova* b;
    b = new Nova[2];
    // Point X
    MoreNova(a);
    // Point Q
    return (0);
}
```

(a) Indicate the number of objects of type `Nova` that exist in memory when the program execution reaches `Point X`.

Answer: ____

(b) Indicate the **change** in the number of objects of type `Nova` that exist in memory and that occurs during program execution between `Point X` and `Point Y`. For example, if 5 more objects exist, write +5. If two fewer objects exist, write -2.

Answer: ____

(c) Indicate the **change** in the number of objects of type `Nova` that exist in memory and that occurs during program execution between `Point Y` and `Point Z`. For example, if 5 more objects exist, write +5. If two fewer objects exist, write -2.

Answer: ____

(d) Indicate the **change** in the number of objects of type `Nova` that exist in memory and that occurs during program execution between `Point Z` and `Point W`. For example, if 5 more objects exist, write +5. If two fewer objects exist, write -2.

Answer: ____

(e) Indicate the **change** in the number of objects of type `Nova` that exist in memory and that occurs during program execution between `Point W` and `Point Q`. For example, if 5 more objects exist, write +5. If two fewer objects exist, write -2.

Answer: ____

Hint: Draw a picture!

Question 12. (7 marks). *Constructors and Stringstreams.*

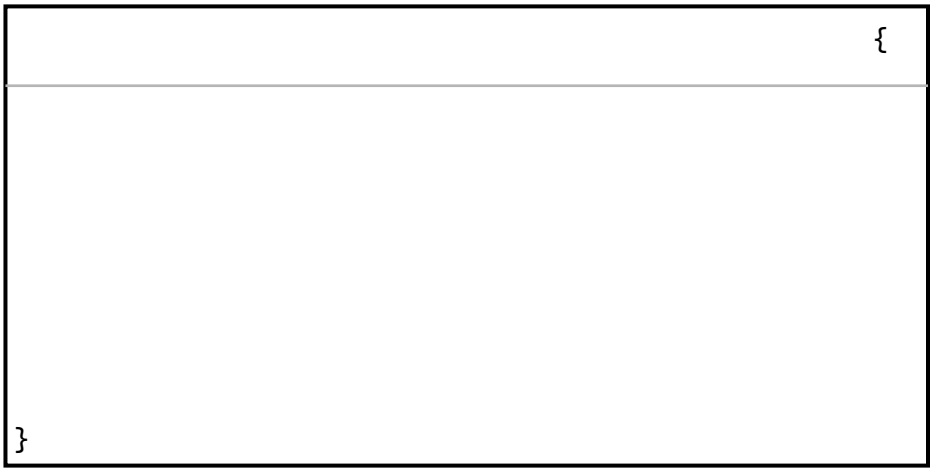
Consider the following declaration of a class that represents a time of day, in the file `Time.h`. You may assume that this class is implemented correctly and is error-free.

```
#include <iostream>
using namespace std;

class Time {
   private:
      int hour, minute, second;

   public:
      Time (int h, int m, int s);

      int getHour() const;
      int getMinute() const;
      int getSecond() const;

      void setHour(int h);
      void setMinute(int m);
      void setSecond(int s);
};
```

We wish to be able to write the following code in the function main.

```
#include <Time.h" // This includes the above definition of Time
#include <string>
#include <sstream>

using namespace std;

int main () {
    string s;
    s = "13 5 10"; // hour is 13, minute is 5, second is 10
                   // assume no errors in s
    Time now(s);

    return (0);
}
```

The above code requires that one member function be added to the class. Write this function in the space below. Your answer should not exceed a few lines of code.



Question 13. (8 marks). *Constructors and Destructor.*

Consider the following definition/implementation of a class called `Box` that appears in the file: `Box.h`. You can assume that the file has no compile-time errors.

```
#include <iostream>
using namespace std;

class Box {
  private:
    int ID;

  public:
    Box () {
      ID = 0;
      cout << "Constructor 1 " << ID << endl;
    }

    Box (int id) {
      ID = id;
      cout << "Constructor 2 " << ID << endl;
    }

    Box (const Box & s) {
      ID = s.ID;
      cout << "Constructor 3 " << ID << endl;
    }

    ~Box() { cout << "Destructor " << endl;}

    Box& operator=(Box & rhs) {
      cout << "Operator= " << ID << endl;
      ID = rhs.ID;
      return (*this);
    }

    int  getID() const { return ID; }
    void setID(int id) { ID = id; }
};
```

The following is a main program that uses the above class. You should assume it compiles and runs correctly.

```
#include <iostream>
using namespace std;

#include "Box.h"

int getID (Box & s) {s.setID(9); return s.getID();}
int getBoxID(Box s) {s.setID(3); return s.getID();}
void setID (Box s) {s.setID(7);}

Box square;

int main () {
  Box rectangle(5);
  Box cube[2];
  Box* hexagon[2];
  hexagon[0] = new Box(rectangle);
  hexagon[1] = hexagon[0];
  cube[0] = *hexagon[0];
  cube[1] = *hexagon[1];
  cout << getBoxID(cube[0]) << endl;
  cout << cube[0].getID() << endl;
  cout << getID(cube[0]) << endl;
  cout << cube[0].getID() << endl;
  setID(cube[1]);
  cout << cube[1].getID() << endl;

  return 0;
}
```

What is the output of the program? Select one of the answers from the table below. Do **NOT** circle an answer in the table. Put your answer in the box below.

Write one of `A`, `B`, `C`, `D`, `E`, or `F` here:

Note: There is only one correct answer that receives the full mark. However, incorrect answers do get part marks (some get more than others and some get none).

A	Constructor 2 5 Constructor 1 0 Constructor 1 0 Constructor 3 5 Operator= 0 Operator= 0 Constructor 3 5 3 Destructor 5 9 9 Constructor 3 5 Destructor 5 Destructor Destructor Destructor	D	Constructor 1 0 Constructor 2 5 Constructor 1 0 Constructor 1 0 Constructor 3 5 Operator= 0 Operator= 0 Constructor 3 5 3 Destructor 5 Constructor 3 5 9 Destructor 5 Constructor 3 5 Destructor 5 Destructor Destructor Destructor Destructor
B	Constructor 1 0 Constructor 2 5 Constructor 1 0 Constructor 1 0 Constructor 3 5 Operator= 5 Operator= 0 Operator= 0 Constructor 3 5 3 Destructor 5 9 9 Constructor 3 5 Destructor 5 Destructor Destructor Destructor Destructor	E	Constructor 1 0 Constructor 2 5 Constructor 1 0 Constructor 1 0 Constructor 3 5 Operator= 0 Operator= 0 3 3 9 9 Constructor 3 5 Destructor 5 Destructor Destructor Destructor Destructor

C	Constructor 1 0 Constructor 2 5 Constructor 1 0 Constructor 1 0 Constructor 3 5 Operator= 0 Operator= 0 Constructor 3 5 3 Destructor 5 9 9 Constructor 3 5 Destructor 5 Destructor Destructor Destructor Destructor	F	Constructor 2 5 Constructor 1 0 Constructor 1 0 Constructor 3 5 Operator= 0 Operator= 0 Constructor 3 5 3 Destructor 5 9 9 7 Destructor Destructor Destructor

Question 14. (9 marks). *Operator Overloading*.

The following class is used to create objects that represent ordinary fractions `n/d`, consisting of a numerator `n` and a denominator `d`.

```
#include <iostream>
using namespace std;

class Fraction {
   private:
      int numerator;
      int denominator;

   public:
      Fraction(int num, int denm);
      int getNumerator();
      int getDenominator();
      void setNumerator(int num);
      void setDenominator(int denm);
      void print();
};

Fraction::Fraction(int num, int denm) {
      numerator = num;
      // Should check that denm is not 0, but ignore for now
      denominator = denm;
}

int Fraction::getNumerator() {
      return (numerator);
}

int Fraction::getDenominator() {
      return (denominator);
}

void Fraction::setNumerator(int num) {
      numerator = num;
}

void Fraction::setDenominator(int denm) {
      // Should check that denm is not 0, but ignore for now
      denominator = denm;
}

void Fraction::print() {
      cout << numerator << "/" << denominator << endl;
}
```

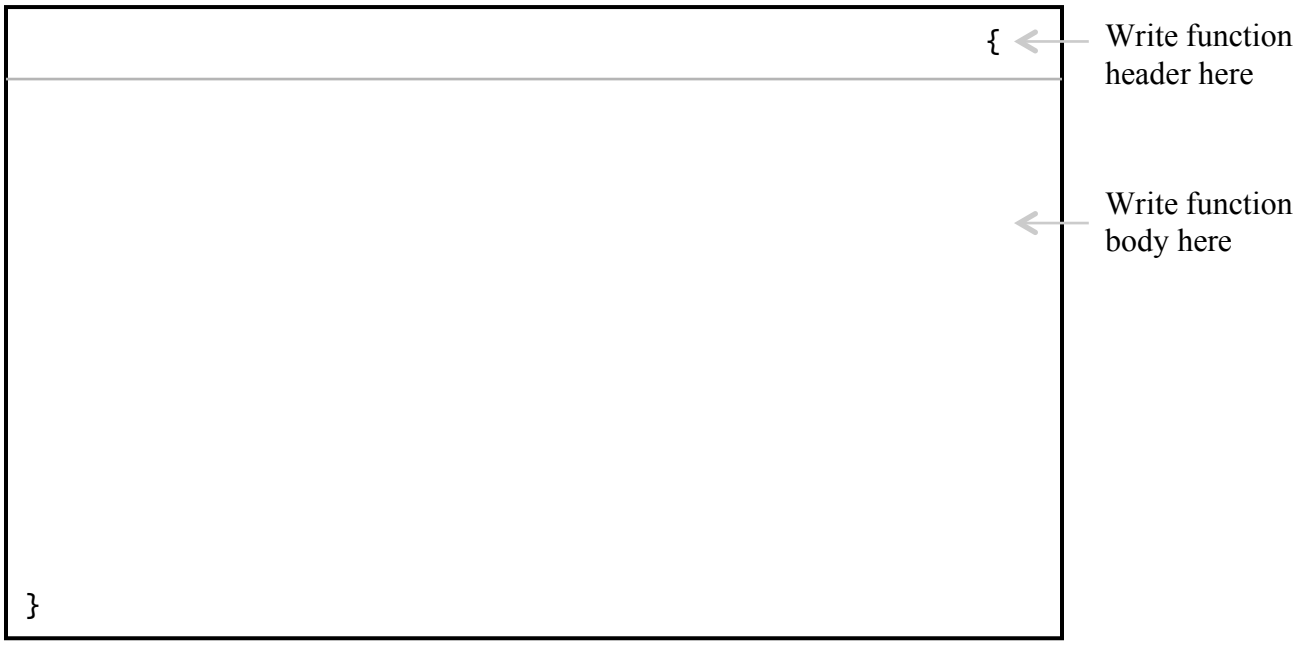
We wish to overload the "*" operator for the `Fraction` class to be able to write code like this in a non-member function (say `main`):

```
Fraction X(1,5);
Fraction Y(4,6);
   :
.. = X * Y;   // The first multiply operation
.. = X * 2;   // The second multiply operation
```

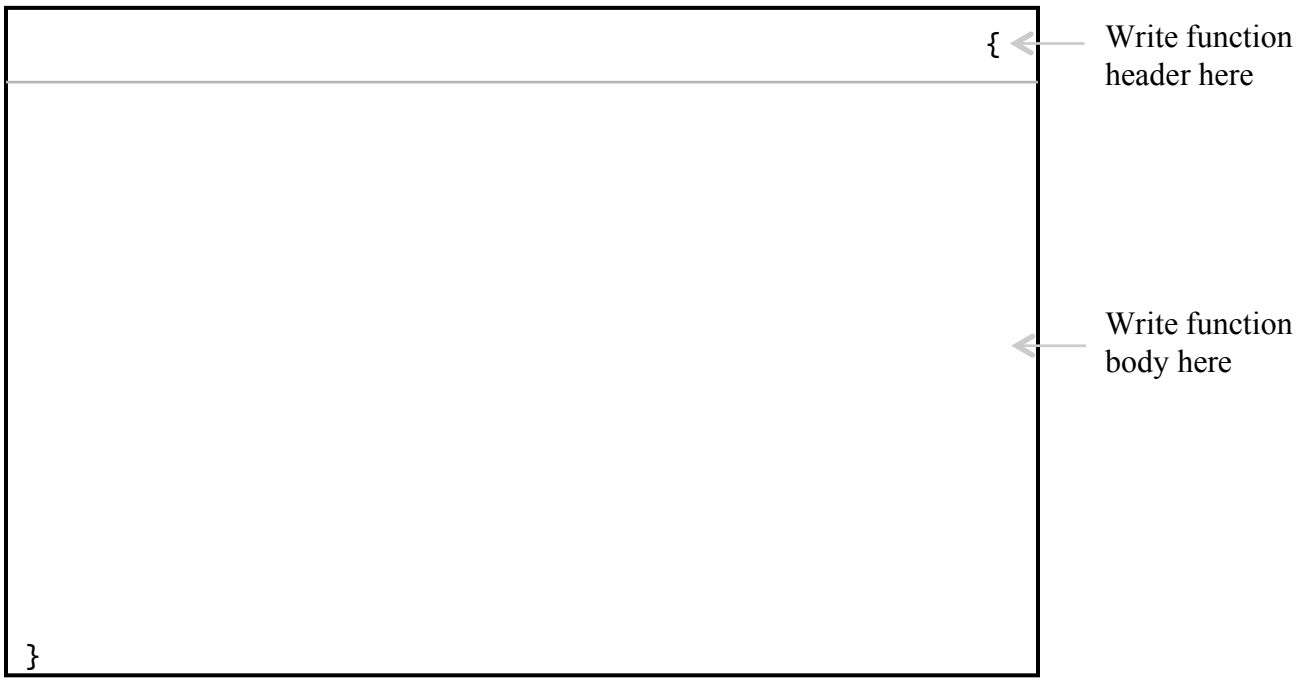
For example, if X represents "1/5" and Y represents "4/6" then `X * Y` results in an object that represents "4/30", while `X * 2` results in an object that represents "2/10". That is, both the numerator and denominator are multiplied by 2.

Write the implementation of the two overloaded operator functions as members of the class `Fraction`. Clearly show the function header and its body.

(a) (6 marks). Overload the multiplication operator `*` as a member of the class `Fraction` to be able to perform the first multiply operation (see comment above). Write your answer in the box below. Be sure to indicate both the header and the body of the method. You need not worry about using `const` modifiers.



(b) (3 marks). Overload the multiplication operator `*` as a member of the class `Fraction` to be able to perform the second multiply operation (see comment above). Write your answer in the box below. Be sure to indicate both the header and the body of the method. You need not worry about using `const` modifiers.



This Page is Intentionally Left Blank – Use for Rough Work