```
Name: Jude Onyia
Student ID: V00947095
Email: judeonyia10@gmail.com
Course: ECE596C
Section: T01

Assignment ID: cpp_basics
Assignment Title: C++ Basics

Submission Source: https://github.com/uvic-seng475-2020-05/cpp_basics-JudeOnyia.
git
Commit ID: 0e3feb72b69883aa24dcfb106a593b9975128439

Submitted Files
===============

drwxrwxr-x    4096 2020-05-22 19:49 ./app
-rw-rw-r--    2243 2020-05-22 19:49 ./app/test_random.cpp
-rw-rw-r--    6163 2020-05-22 19:49 ./app/test_rational.cpp
-rw-rw-r--     343 2020-05-22 19:49 ./CMakeLists.txt
-rw-rw-r--     140 2020-05-22 19:49 ./IDENTIFICATION.txt
drwxrwxr-x    4096 2020-05-22 19:49 ./include
drwxrwxr-x    4096 2020-05-22 19:49 ./include/ra
-rw-rw-r--    2391 2020-05-22 19:49 ./include/ra/random.hpp
-rw-rw-r--    6835 2020-05-22 19:49 ./include/ra/rational.hpp
drwxrwxr-x    4096 2020-05-22 19:49 ./lib
-rw-rw-r--    1287 2020-05-22 19:49 ./lib/random.cpp
-rw-rw-r--  391622 2020-05-22 19:49 ./README.pdf

Results
=======

Package          Operation Target           Status
nonprog          generate  ---              OK (0.0s)
random_orig      generate  ---              OK (0.1s)
random_orig      configure ---              OK (0.7s)
random_orig      build     test_random      FAIL (2 0.1s 2L)
random_sane      generate  ---              OK (0.2s)
random_sane      configure ---              OK (0.9s)
random_sane      build     test_random      OK (1.1s)
rational_orig    generate  ---              OK (0.1s)
rational_orig    configure ---              OK (0.6s)
rational_orig    build     test_rational    FAIL (2 0.1s 2L)
rational_sane    generate  ---              OK (0.2s)
rational_sane    configure ---              OK (0.7s)
rational_sane    build     test_rational    OK (1.3s)

Normally, an operation is indicated as having a status of either "OK" or
"FAIL".  A status of "?" indicates that the operation could not be performed
for some reason (e.g., due to an earlier error or being a manual step).
The time (in seconds) required for an operation is denoted by an expression
consisting of a number followed by the letter "s" (e.g., "5.0s").
In the case of a test that consists of multiple test cases, the number of
failed test cases and total number of test cases is expressed as a fraction
(e.g., "10/50" means 10 test cases failed out of 50 test cases in total).
The length (in lines) of the log file generated by an operation is denoted by
an expression consisting of a number followed by the letter "L" (e.g., "10L").
To ascertain the reason for the failure of an operation, check the contents
of the log file provided.
```

```
Legend
======

Package: nonprog
    Nonprogramming exercises

Package: random_orig
    The code as originally submitted by the student.
Build target: test_random
    Build the test_random program.

Package: random_sane
    Code with modifications to perform API sanity checking.
Build target: test_random
    Build the test_random program.

Package: rational_orig
    The code as originally submitted by the student.
Build target: test_rational
    Build the test_rational program.

Package: rational_sane
    Code with modifications to perform API sanity checking.
Build target: test_rational
    Build the test_rational program.
```

```
1  gmake: *** No rule to make target `test_random'.  Stop.
2  ERROR: build failed to generate executable test_random
```

```
1  gmake: *** No rule to make target `test_rational'.  Stop.
2  ERROR: build failed to generate executable test_rational
```

```
commit b985c965a338d89c78c23abef0753310181f7f60
Author: JudeOnyia <60678029+JudeOnyia@users.noreply.github.com>
Date:   Sat May 16 21:50:59 2020 -0700

    First commit. Added IDENTIFICATION text file

commit 4ff2d9e74af1f94ec6313fc01c20c8c8e3e27d16
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Mon May 18 13:12:36 2020 -0700

    B-1 part a to part e (detailed description below)
    1) Set up the files to contain the linear_congruential_generator class
    2) Created the type member, int_type
    3) Created the constructor
    4) Created the static member function, default_seed
    5) Created data members, a_, c_, m_ and s_
    6) Created static data member, seed_

commit 7eca19d8d2e76ce9cd63378701749c9bb635e3bf
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Mon May 18 13:51:49 2020 -0700

    Moved class definition from hpp file to cpp file, fixed the
    static function default_seed, and removed static variable seed_.

commit 6c3e60a42aad47c86ffe70982d73c3d5a64a958e
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Mon May 18 15:26:42 2020 -0700

    B-1 part f to part i (detailed description below)
    1) created the multiplier, increment and modulus functions
    2) created the seed function to restart the sequence generation process
    3) add a member data to indicate how many next positions should be
       discarded in the generated sequence. This defaults to zero
    4) created the discard function to set the discard member data
    5) overloaded the operator() function to advance the generator to the
       next position, and skip positions that are to be discarded

commit 9c8bd027d1dc11a56e090004d1ba352b4367890d
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Mon May 18 19:16:06 2020 -0700

    B-1 part m to part o (detailed description below)
    1) created the min and max member functions
    2) overloaded the equality and inequality operators
    3) provided a stream inserter (non member function)
    4) enveloped the class and stream inserter with namespace ra::random

commit 26b588789ebd411370795830191f65b188d68c3e
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Tue May 19 01:14:59 2020 -0700

    Attempt on test code for lcg class.

commit 3d58a7f5c52fe5f1cac773182784a3bba2a349e3
Author: JudeOnyia <60678029+JudeOnyia@users.noreply.github.com>
Date:   Tue May 19 01:41:40 2020 -0700

    Fixed the random.hpp file to have the proper declarations, and
    modifies the random.cpp file to have definitions of some member functions
    and the non-member function (operator<<).

```

```
commit 9793c850025d636b22bc0b4fc20624eab8dd0f13
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Tue May 19 02:13:19 2020 -0700

    Some error correction made of random.hpp and the test_random.cpp

commit ed7deaa5cea91373489078fd2cd3a38a5d18f16c
Author: JudeOnyia <60678029+JudeOnyia@users.noreply.github.com>
Date:   Tue May 19 19:47:55 2020 -0700

    Corrected the definition of the seed member function and the operator()

commit b78d4a953eeabfd9cc5cf3e5866afb36e2d2378b
Author: JudeOnyia <60678029+JudeOnyia@users.noreply.github.com>
Date:   Tue May 19 20:04:24 2020 -0700

    Moved the constructor and the stream inserter definitions back to the header
 file

commit 27cd751979710a2e089c05cedc0f1df4332eac70
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Tue May 19 20:49:00 2020 -0700

    Code Finally Builds. Moved stream inserter definition back to
    random.cpp, removed the const prefix in stream inserter.

commit af187b18f31b16877d39b41292d52ce67087077e
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Wed May 20 01:58:14 2020 -0700

    Completed draft of the test of every member and non member function

commit 2d6841b73b55efdf874307fa8e8d7ec305e31d0c
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Wed May 20 02:21:52 2020 -0700

    Moved constructor to random.cpp

commit e76162cfa44cc606be634bb1f86470d3de21128a
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Wed May 20 15:44:11 2020 -0700

    Began B2 Parts a to h (detailed below)
    1) created the template class rational
    2) created the default constructor
    3) created the two parameter constructor
    4) created the numerator and denominator member functions

commit 56ef8368c9372dd7fc68ff3d3c6a54d3a4ea97f4
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Wed May 20 17:04:06 2020 -0700

    Wrote set up for compound assignment operators

commit 336f8371c02d5d8c31f0e09d20fbac69f6cd4d05
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Wed May 20 18:18:27 2020 -0700

    1) Created the truncation function
    2) Wrote test for the default constructor
    3) Wrote test for constructor with single argument
    4) Wrote test for constructor with 2 arguments
```

```
    5) Wrote test for truncation function

commit a3443247cb741d1417acf72568bdbf6b0fd88c4c
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Wed May 20 19:20:23 2020 -0700

    1) Wrote is_integer function
    2) Tested is_integer function

commit 5259695fc4569c708380890aff19c477f40c2b1a
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Wed May 20 19:40:18 2020 -0700

    1) Wrote operator oveload for Not(!) operator
    2) Tested Not(!) operator overload

commit 81dea77023c36e9e44073e4a565c44f661c16674
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Wed May 20 20:11:56 2020 -0700

    1) Wrote the Equality(==) and Inequality(!=) operator overloads
    2) Tested these operator overloads

commit 1b9e521fe3e52ec3ae9281ecce871b470fa79c02
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Wed May 20 21:04:59 2020 -0700

    1) Wrote the operator overloads for: <, >, <=, >=
    2) Tested these opertor overloads

commit 944cd522cd5ede4c7ac8faa1b9530669cbed38ae
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Wed May 20 23:31:28 2020 -0700

    1) Wrote code for maintaining reduced form of rational number
    2) Wrote code for ensuring that denominator is not negative
    3) Tested both code

commit 0e547c08ff83c80e76d4ca9ca9761317ba487486
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Wed May 20 23:59:36 2020 -0700

    1) Wrote condition for when the denominator is zero
    2) Tested this condition

commit 34d8990468b542e4574f8c13a36dc4dc55b7294a
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Thu May 21 00:52:45 2020 -0700

    1) Fixed the truncation function
    2) Wrote operator oveload for prefix increment and decrement
    3) Tested operato oveloads

commit e9433fdab5c078db150a3b6a98a86f3df3701b9e
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Thu May 21 01:06:11 2020 -0700

    1) Wrote operator overload of postfix increment and decrement
    2) Tested these operator oveloads

commit e010f2b773d186879550d3bfe3f7f4c980737195
Author: Jude Onyia <judeonyia10@gmail.com>
```

```
Date:   Thu May 21 17:27:22 2020 -0700

    1) Wrote code to turn the numerator and denominator to be whole numbers
       if they weren't.
    2) wrote operator overloads for (+=),(-=),(*=),and (/=)
    3) Tested these operators

commit e0548c8d558cd4ada2bd90c01f6cdee196e84d08
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Thu May 21 19:21:53 2020 -0700

    1) Wrote non-member operator overloads Unary plus(+) and minus(-)
    2) Tested these overloads

commit d445a44110bbe92770b9e072335b7b5233153fbf
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Thu May 21 20:23:09 2020 -0700

    1) Wrote the code for operator oveload of binary add, sub, mult, div
    2) Tested these overloads

commit 9938166711a7217237a7db89686466edd839e740
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Thu May 21 23:25:53 2020 -0700

    1) Wrote Stream Inserter overload and Stream Extractor overload
    2) Tested both overloads

commit 1bcd8abbd4501751ea170a67def4f3d1e4c2f202
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Fri May 22 00:21:46 2020 -0700

    Make sure both the random and rational classes had const correctness

commit ced72293aad6b851b297ed027b22cde116a1aa57
Author: JudeOnyia <60678029+JudeOnyia@users.noreply.github.com>
Date:   Fri May 22 01:44:33 2020 -0700

    Added Fake README.pdf just to test assignment precheck

commit d00bca51cb1849ae3f839288912b9c879f2566a3
Author: JudeOnyia <60678029+JudeOnyia@users.noreply.github.com>
Date:   Fri May 22 14:04:42 2020 -0700

    Added the right README.pdf

commit 5ceed22f6bf97b14db0740bfeccbcd96470460bb
Author: Jude Onyia <judeonyia10@gmail.com>
Date:   Fri May 22 14:17:39 2020 -0700

    Removed the exception in stream extractor of rational.hpp

commit db4937544ca01321325836abb71e7cffd584457c
Author: Jude Onyia <judeonyia@ugls5.ece.uvic.ca>
Date:   Fri May 22 14:31:05 2020 -0700

    Check report

commit 7b152b90b1aae6b185214853ebc2d42a85e9389a
Author: Jude Onyia <judeonyia@ugls5.ece.uvic.ca>
Date:   Fri May 22 14:42:38 2020 -0700

```

```
248     Removed report assigne precheck
249
250 commit 86b90ff76938b2da038b64c34a150cbaaf6b5c2c
251 Author: Jude Onyia <judeonyia@ugls5.ece.uvic.ca>
252 Date:   Fri May 22 14:47:58 2020 -0700
253
254     Check folder
255
256 commit 3a6b6084133a81368eb501623cd9bde59709657e
257 Author: JudeOnyia <60678029+JudeOnyia@users.noreply.github.com>
258 Date:   Fri May 22 14:58:54 2020 -0700
259
260     Check against his test
261
262 commit 0f43774ec13aa5f84925aec6d2ca3e5ad6208b05
263 Author: JudeOnyia <60678029+JudeOnyia@users.noreply.github.com>
264 Date:   Fri May 22 15:09:02 2020 -0700
265
266     changed to consts
267
268 commit 8ab5f4b17920f07531c4e4e145f2fcbc3f482e91
269 Author: JudeOnyia <60678029+JudeOnyia@users.noreply.github.com>
270 Date:   Fri May 22 15:15:08 2020 -0700
271
272     SOmething added
273
274 commit 05caf3bcb3964767225ad4c45ada95dec59c06c5
275 Author: JudeOnyia <60678029+JudeOnyia@users.noreply.github.com>
276 Date:   Fri May 22 15:19:39 2020 -0700
277
278     Fixed more const correctness
279
280 commit 89fd28e397fce89c5e82035dfe101f60a26b01f6
281 Author: JudeOnyia <60678029+JudeOnyia@users.noreply.github.com>
282 Date:   Fri May 22 15:24:49 2020 -0700
283
284     Removed his test case
285
286 commit 8424743864c5b2c92bae384c8bc58ad52339013e
287 Author: Jude Onyia <judeonyia@ugls5.ece.uvic.ca>
288 Date:   Fri May 22 15:31:03 2020 -0700
289
290     Put orgin
291
292 commit 6820f7fe930a7d3bb116d89b4131dea2242d8603
293 Author: JudeOnyia <60678029+JudeOnyia@users.noreply.github.com>
294 Date:   Fri May 22 15:36:06 2020 -0700
295
296     Test mine
297
298 commit 0e3feb72b69883aa24dcfb106a593b9975128439
299 Author: Jude Onyia <judeonyia@ugls5.ece.uvic.ca>
300 Date:   Fri May 22 15:53:18 2020 -0700
301
302     FInal
```

Name: Jude Onyia

Student ID: V00947095

Course: ECE 596C

Due Date: May 22, 2020

Assignment 1: Non – Programming Exercise

8.8 a)

If the tree is balanced and we assume worst case, the asymptotic time complexity of the function is the height of the balanced tree, which is **O(log n)**.

8.8 b)

If the tree is not balanced, assuming worst case of the search for a node with the value and worst case of the imbalance of the tree, the asymptotic time complexity is **O(n)**.

8.9 a)

The source code performs a sequential accumulative sum of the lower triangle of the matrix. From inspecting the source code, it is evident that the elements included in the accumulation consist of half of the matrix excluding the primary diagonal elements (i.e. a(0,0), a(1,1), etc.), plus the primary diagonal elements. Since the code loops over these elements, the asymptotic time complexity is $O(\frac{n^2-n}{2}+n)$, this can be reduced to $\mathbf{O(n^2)}$.

8.9 b)

Since the allocation of memory for the variables created in this function are not dependent on n, assuming the maximum value of type int is greater than n, then the asymptotic space complexity of the function is **O(1)**.

8.10 a)

The asymptotic time complexity of reverse_array_1 is $O(\frac{n}{2})$, this can be reduced to **O(n)**. Assuming the maximum value of type int is greater than n, the asymptotic space complexity is **O(1)**.

8.10 b)

The asymptotic time complexity of reverse_array_2 is **O(n)**. The space complexity is **O(n)** because a vector of size n is created. The assumption here is also that the maximum value of type int is great than n.

8.10 c)

Based on asymptotic complexity analysis, both have the same time complexity, however, reverse_array_1 has a space complexity of O(1) while reverse_array_2 has O(n). Therefore, reverse_array_1 is preferable.

8.12)

We would need to calculate the overall speedup of the program when each of the three parts are optimized.

A) If part A is optimized, the overall speedup of the program is:

$$S_o = \frac{1}{(1 - f_e) + \frac{f_e}{S_e}} = \frac{1}{(1 - 0.05) + \frac{0.05}{10}} = 1.0471$$

B) If part B is optimized, the overall speedup of the program is:

$$S_o = \frac{1}{(1 - f_e) + \frac{f_e}{S_e}} = \frac{1}{(1 - 0.5) + \frac{0.5}{1.05}} = 1.0244$$

C) If part C is optimized, the overall speedup of the program is:

$$S_o = \frac{1}{(1 - f_e) + \frac{f_e}{S_e}} = \frac{1}{(1 - 0.1) + \frac{0.1}{3}} = 1.0714$$

Based on the above calculations, the choice that would yield the most speedup is optimizing part C, therefore, part C should be optimized.

8.13 a)

If we assume the worst case of all bits having the value 1 (or even just the most significant bit having the value 1), the while loops will iterate until the most significant bit of value 1 has been checked. Hence, it will iterate for the bit-length of the integer. The number of bits of the integer is $\log_2(n)$, rounded up. Therefore, the asymptotic time complexity is **O(log n)**. The asymptotic space complexity is **O(1)** because if the number of bits used for n is changed, the only memory affected is that of n.

8.13 b)

The code below is an implementation of the algorithm derived from [1].

```
unsigned int hamming_2(unsigned int n){
    unsigned int total_bit_num = sizeof(int) * CHAR_BITS; // Number of bits in n
    unsigned int partition_1 = (~(unsigned int)0) / 3; // Binary 01010101
    unsigned int partition_2 = (~(unsigned int)0) / 5; // Binary 00110011
    unsigned int partition_4 = (~(unsigned int)0) / 17; // Binary 00001111

    n -= (n >> 1) & partition_1;  //Count the ones of each 2 bits and
                                  //replace those 2 bits with result
```

```
    n = (n & partition_2) + ((n >> 2) & partition_2); //Count the ones of each 4 bits
                                                      //and replace those 4 bits with result
    n = (n + (n >> 4)) & partition_4; // Count the ones of each 8 bits and
                                      //replace those 8 bits with result

    if(total_bit_num > 8)  n += n >>  8; //move result of each 16 bits into lowest 8 bits
    if(total_bit_num > 16) n += n >> 16; //move result of each 32 bits into lowest 8 bits
    if(total_bit_num > 32) n += n >> 32; //move result of each 64 bits into lowest 8 bits
    return n & 0x7F; // bit AND with decimal number 127 will keep the 8 bits
}
```

The advantage of the algorithm is that it's asymptotic time complexity is **O(1)**, less than hamming_1's complexity of O(log n). The disadvantage is that it requires more space in memory than hamming_1.

8.13 c)

The reasoning behind using asymptotic complexity is to have a sense of the effect of problem size on the performance of the program as the problem size increases to relatively huge amount. The asymptotic analysis is necessary to calculate the rate of program's performance and memory requirement as the problem size increases.

Reference

[1] Joel Yliluoma, WP2 - Nifty Revised, without multipliations, Bit-counting algorithms, 2013. https://bisqwit.iki.fi/source/misc/bitcounting/

```
1  # Specify Minimum Required Version
2  cmake_minimum_required(VERSION 3.1 FATAL_ERROR)
3
4  # Specify Project and Language
5  project(random_and_rational LANGUAGES CXX)
6
7  # Set Include Directory
8  include_directories(include)
9
10 # Add Executable Program
11 add_executable(random app/test_random.cpp lib/random.cpp)
12 add_executable(rational app/test_rational.cpp)
```

```
1  #ifndef random_hpp
2  #define random_hpp
3  #include <iostream>
4  namespace ra::random{
5      class linear_congruential_generator {
6      public:
7          typedef unsigned long long int int_type; // type member
8          static int_type default_seed(){ return (int_type)1;} // Function to retu
   rn default seed of one for all objects
9
10         // Constructor that initializes the multiplier, increment and modulus. S
   eed is optional argument.
11         linear_congruential_generator(int_type a, int_type c, int_type m, int_ty
   pe s = default_seed());
12
13         const int_type multiplier() const { return a_;} // Function to return mu
   ltiplier value
14         const int_type increment() const { return c_;} // Function to return inc
   rement value
15         const int_type modulus() const { return m_;} // Function to return modul
   us value
16         const int_type position() const { return x_;} // Function to return the
   current position in the sequence
17
18         // Function to restarts the sequence generation process with a new seed
   value
19         void seed(int_type s);
20
21         // Operator to advance the generator to the next position in the sequenc
   e
22         // with consideration to the number of positions to be discarded
23         int_type operator()();
24
25         // Function to discard the next n numbers in the generated sequence
26         void discard(unsigned long long n){ n_ = n; }
27
28         const int_type min() const { return c_==(int_type)0? (int_type)1 : (int_
   type)0; } // Function to get the smallest value
29         const int_type max() const { return m_-(int_type)1; } // Funtion to get
   the largest value in sequence
30
31         // Operator to test two linear_congruential_generator objects for equali
   ty
32         bool operator==(const linear_congruential_generator& obj) const {
33             return (a_==obj.multiplier() && c_==obj.increment()  && m_==obj.modu
   lus()  && x_==obj.position());
34         }
35
36         // Operator to test two linear_congruential_generator objects for inequa
   lity
37         bool operator!=(const linear_congruential_generator& obj) const {
38                 return !(a_==obj.multiplier() && c_==obj.increment()  && m_==obj
   .modulus()  && x_==obj.position());
39         }
40
41
42     private:
43         int_type a_; // multiplier
44         int_type c_; // increment
45         int_type m_; // modulus
46         int_type x_; // current position in the generated sequence
47         unsigned long long n_ = (unsigned long long)0; // number of positions to
```

```
     discard in the sequence
48      };
49
50      // Stream inserter
51      std::ostream& operator<<(std::ostream& outStream, const linear_congruential_
    generator& objA);
52  }
53  #endif
```

```cpp
#include <iostream>
#include "ra/random.hpp"

namespace ra::random {
        typedef linear_congruential_generator::int_type int_type;

        // Constructor that initializes the multiplier, increment and modulus. Seed is optional argument.
        linear_congruential_generator::linear_congruential_generator(int_type a, int_type c, int_type m, int_type s){
            a_ = a;
            c_ = c;
            m_ = m;
            if( (c_ % m_)==(int_type)0 && (s % m_)==(int_type)0 ) x_ = (int_type)1;
            else x_ = s;
        }

        // Function to restarts the sequence generation process with a new seed value
        void linear_congruential_generator::seed(int_type s){
            if( (c_ % m_)==(int_type)0 && (s % m_)==(int_type)0 ) x_ = (int_type)1;
            else x_ = s;
            n_ = (unsigned long long)0;
        }

        // Operator to advance the generator to the next position in the sequence
        // with consideration to the number of positions to be discarded
        int_type linear_congruential_generator::operator()(){
            ++n_;
            do{
                x_ = (a_ * x_ + c_) % m_;
                --n_;
            } while(n_);
            return x_;
        }

        // Stream inserter
        std::ostream& operator<<(std::ostream& outStream, const linear_congruential_generator& objA){
            outStream << objA.multiplier() << " " << objA.increment() << " " << objA.modulus() << " " << objA.position();
            return outStream;
        }


}
```

```cpp
#include "ra/random.hpp"
#include <iostream>
#include <random>

int main(){

    typedef ra::random::linear_congruential_generator::int_type int_type;
    using std::cout;
    using std::endl;

    // Test class against linear congruential engine in standard library
    // Test constructor with no seed input
    // Test the operator() and the operator<<
    ra::random::linear_congruential_generator obj_mine(14,5,29);
    std::linear_congruential_engine<std::uint_fast32_t,14,5,29> obj_theirs;
    obj_mine();
    obj_theirs();
    //cout << "lc generator object: " << obj_mine << endl;
    //cout << "lc engine current state: " << obj_theirs << endl;

    // Compare their minimum and maximum
    //cout << "lc generator min value: " << obj_mine.min() << endl;
    //cout << "lc engine min value: " << obj_theirs.min() << endl;
    //cout << "lc generator max value: " << obj_mine.max() << endl;
    //cout << "lc engine max value: " << obj_theirs.max() << endl;

    // Test constructor with seed input
    // Test seed() member function
    // Test operator== and operator!=
    ra::random::linear_congruential_generator obj_mine_A(97,41,300,77);
    //cout << "lc generator object (seed must be 77): " << obj_mine_A << endl;
    //obj_mine_A.seed(259);
    //cout << "lc generator object (seed change to 259): " << obj_mine_A << endl
;
    obj_mine_A.seed(77);
    ra::random::linear_congruential_generator obj_mine_B(97,41,300,77);
    ra::random::linear_congruential_generator obj_mine_C(20,58,300,77);
    //cout << "lc generator equality check (Must be true): " << (obj_mine_A==obj
_mine_B) << endl;
    //cout << "lc generator equality check (Must be false): " << (obj_mine_A==ob
j_mine_C) << endl;
    //cout << "lc generator inequality check (Must be false): " << (obj_mine_A!=
obj_mine_B) << endl;
    //cout << "lc generator inequality check (Must be true): " << (obj_mine_A!=o
bj_mine_C) << endl;

    // Test the discard member function
    for(int i=0; i<90; ++i){
        obj_mine_A();
    }
    obj_mine_B.discard(90);
    //cout << "lc generator discard function check (Must be true): " << (obj_min
e_A()==obj_mine_B()) << endl;

    // Test condition when increment and seed are both zero
    ra::random::linear_congruential_generator obj_mine_D(20,0,300,0);
    //cout << "lc generator seed (Must be 1): " << obj_mine_D << endl;

    return 0;


```

```
57 }
```

```
1  #ifndef rational_hpp
2  #define rational_hpp
3  #include <iostream>
4  #include <algorithm>
5  #include <string>
6  #include <sstream>
7  namespace ra::math{
8  template<class T>
9  class rational {
10     public:
11         typedef T int_type;
12
13         // Function to reduce the form of the rational number
14         void reduce_form(){
15             long long the_gcd = std::__gcd((long long)n_,(long long)d_);
16             n_ = (int_type)( (long long)n_ / the_gcd ); // Also make numerator a
    whole number;
17             d_ = (int_type)( (long long)d_ / the_gcd ); // Also make denominator
    a whole number
18         }
19
20         // Function to Prevent denominator from having zero or negative value
21         void denominator_handle(){
22             if(d_ == (int_type)0){
23                 n_ = std::numeric_limits<int_type>::max();
24                 d_ = (int_type)1;
25             }
26             if(d_ < (int_type)0) { d_ = d_ * (int_type)(-1); n_ = n_ * (int_type
    )(-1); }
27         }
28
29         // Default constructor sets rational number to 0
30         rational(){
31             n_ = (int_type)0;
32             d_ = (int_type)1;
33         }
34
35         // Constructor to specify numerator and denominator values
36         rational(int_type n, int_type d = (int_type)1){
37             n_ = n;
38             d_ = d;
39             reduce_form();
40             denominator_handle();
41         }
42
43         const int_type numerator() const { return n_; } // Function to return th
    e numerator value
44         const int_type denominator() const { return d_; } // Function to return
    the denominator value
45
46         // Operator for compound addition (+=)
47         rational& operator+=(const rational& obj){
48             n_ = (n_ * obj.denominator()) + (obj.numerator() * d_);
49             d_ = d_ * obj.denominator();
50             reduce_form();
51             return *this;
52         }
53
54         // Operator for compound subtraction (-=)
55         rational& operator-=(const rational& obj){
56             n_ = (n_ * obj.denominator()) - (obj.numerator() * d_);
57             d_ = d_ * obj.denominator();
```

```
58              reduce_form();
59              return *this;
60          }
61
62          // Operator for compound multiplication (*=)
63          rational& operator*=(const rational& obj){
64              n_ = n_ * obj.numerator();
65              d_ = d_ * obj.denominator();
66              reduce_form();
67              return *this;
68          }
69
70          // Operator for compound division (/=)
71          rational& operator/=(const rational& obj){
72              n_ = n_ * obj.denominator();
73              d_ = d_ * obj.numerator();
74              reduce_form();
75              denominator_handle();
76              return *this;
77          }
78
79          // Function for rounding the rational number towards zero (discard fract
    ional part)
80          const int_type truncate() const {
81              return (int_type)((long long)(n_ / d_));
82          }
83
84          // Function to check if rational number is an integer
85          bool is_integer() const {
86              return ( d_==(int_type)1 );
87          }
88
89          // Operator to check if a rational number is zero (!)
90          bool operator!() const {
91              return (n_==(int_type)0);
92          }
93
94          // Operator to check equality of rational numbers (==)
95          bool operator==(const rational& obj) const {
96              return ( (n_/d_) == (obj.numerator()/obj.denominator()) );
97          }
98
99          // Operator to check inequality of rational numbers (!=)
100         bool operator!=(const rational& obj) const {
101             return ( (n_/d_) != (obj.numerator()/obj.denominator()) );
102         }
103
104         // Operator to check less than of rational numbers (<)
105         bool operator<(const rational& obj) const {
106             return ( (n_/d_) < (obj.numerator()/obj.denominator()) );
107         }
108
109         // Operator to check greater than of rational numbers (>)
110         bool operator>(const rational& obj) const {
111             return ( (n_/d_) > (obj.numerator()/obj.denominator()) );
112         }
113
114         // Operator to check less than or equals to of rational numbers (<=)
115         bool operator<=(const rational& obj) const {
116             return ( (n_/d_) <= (obj.numerator()/obj.denominator()) );
117         }
118
```

```
119         // Operator to check greater than or equals to of rational numbers (>=)
120         bool operator>=(const rational& obj) const {
121             return ( (n_/d_) >= (obj.numerator()/obj.denominator()) );
122         }
123 
124         // Operator to perform prefix increment (++obj)
125         rational& operator++(){
126             n_ = n_ + d_;
127             return *this;
128         }
129 
130         // Operator to perform prefix and decrement (--obj)
131         rational& operator--(){
132             n_ = n_ - d_;
133             return *this;
134         }
135 
136         // Operator to perform postfix increment (obj++)
137         rational operator++(int){
138             rational<int_type> obj_copy(n_,d_);
139             n_ = n_ + d_;
140             return obj_copy;
141         }
142 
143         // Operator to perform postfix decrement (obj--)
144         rational operator--(int){
145             rational<int_type> obj_copy(n_,d_);
146             n_ = n_ - d_;
147             return obj_copy;
148         }
149 
150     private:
151         int_type n_; // Numerator
152         int_type d_; // Denominator
153 };
154 
155 // Operator to perform Unary plus (+)
156 template<class int_type>
157 rational<int_type> operator+(const rational<int_type>& obj){
158     return rational<int_type>(+(obj.numerator()),obj.denominator());
159 }
160 
161 // Operator to perform Unary minus (-)
162 template<class int_type>
163 rational<int_type> operator-(const rational<int_type>& obj){
164     return rational<int_type>(-(obj.numerator()),obj.denominator());
165 }
166 
167 // Operator to perform Binary addition (+)
168 template<class int_type>
169 rational<int_type> operator+(const rational<int_type>& obj_A, const rational<int
    _type>& obj_B){
170     int_type n_result = (obj_A.numerator() * obj_B.denominator()) + (obj_A.denom
    inator() * obj_B.numerator());
171     int_type d_result = obj_A.denominator() * obj_B.denominator();
172     return rational<int_type>(n_result,d_result);
173 }
174 
175 // Operator to perform Binary subtraction (-)
176 template<class int_type>
177 rational<int_type> operator-(const rational<int_type>& obj_A, const rational<int
    _type>& obj_B){
```

```
178     int_type n_result = (obj_A.numerator() * obj_B.denominator()) - (obj_A.denom
    inator() * obj_B.numerator());
179     int_type d_result = obj_A.denominator() * obj_B.denominator();
180     return rational<int_type>(n_result,d_result);
181 }
182
183 // Operator to perform Binary multiplication (*)
184 template<class int_type>
185 rational<int_type> operator*(const rational<int_type>& obj_A, const rational<int
    _type>& obj_B){
186     int_type n_result = obj_A.numerator() * obj_B.numerator();
187     int_type d_result = obj_A.denominator() * obj_B.denominator();
188     return rational<int_type>(n_result,d_result);
189 }
190
191 // Operator to perform Binary division (/)
192 template<class int_type>
193 rational<int_type> operator/(const rational<int_type>& obj_A, const rational<int
    _type>& obj_B){
194     int_type n_result = obj_A.numerator() * obj_B.denominator();
195     int_type d_result = obj_A.denominator() * obj_B.numerator();
196     return rational<int_type>(n_result,d_result);
197 }
198
199 // Stream Inserter
200 template<class int_type>
201 std::ostream& operator<<(std::ostream& outStream, const rational<int_type>& obj)
    {
202     outStream << obj.numerator() << "/" << obj.denominator();
203     return outStream;
204 }
205
206 // Stream Extractor
207 template<class int_type>
208 std::istream& operator>>(std::istream& inStream, rational<int_type>& obj){
209     std::string the_input;
210     std::getline(inStream,the_input);
211     std::istringstream iss(the_input);
212     std::string n, d;
213     std::getline(iss,n,'/');
214     std::getline(iss,d);
215     long long n_l = std::stoll(n);
216     long long d_l = std::stoll(d);
217     if((std::to_string(n_l) + '/' + std::to_string(d_l))!=the_input){
218         inStream.setstate(std::ios_base::failbit);
219     }
220     obj = rational<int_type>((int_type)n_l,(int_type)d_l);
221     return inStream;
222 }
223
224
225
226 }
227 #endif
```

```cpp
#include "ra/rational.hpp"
#include <iostream>

#include <string>
#include <sstream>

int main(){
    using std::cout;
    using std::endl;

    ra::math::rational<double> obj_A;
    //cout << "1) Test default constructor" << endl;
    //cout << "   Numerator: " << obj_A.numerator() << endl;
    //cout << "   Denominator: " << obj_A.denominator() << endl << endl;

    ra::math::rational<float> obj_B(-56);
    //cout << "2) Test constructor with single parameter" << endl;
    //cout << "   Numerator: " << obj_B.numerator() << endl;
    //cout << "   Denominator: " << obj_B.denominator() << endl << endl;

    ra::math::rational<double> obj_C(31488,117);
    //cout << "3) Test constructor with double parameter and trunction function"
 << endl;
    //cout << "   Numerator: " << obj_C.numerator() << endl;
    //cout << "   Denominator: " << obj_C.denominator() << endl;
    //cout << "   Truncated value: " << obj_C.truncate() << endl << endl;

    ra::math::rational<double> obj_D(48,-4);
    //cout << "4) Test is_integer function" << endl;
    //cout << "   Numerator: " << obj_D.numerator() << endl;
    //cout << "   Denominator: " << obj_D.denominator() << endl;
    //cout << "   is_integer: " << obj_D.is_integer() << endl;
    //cout << "   Numerator: " << obj_C.numerator() << endl;
    //cout << "   Denominator: " << obj_C.denominator() << endl;
    //cout << "   is_integer: " << obj_C.is_integer() << endl << endl;

    ra::math::rational<double> obj_E(0,-4);
    //cout << "5) Test the Not(!) operator" << endl;
    //cout << "   Numerator: " << obj_E.numerator() << endl;
    //cout << "   Not(!) operator: " << !obj_E << endl;
    //cout << "   Numerator: " << obj_D.numerator() << endl;
    //cout << "   Not(!) operator: " << !obj_D << endl << endl;

    ra::math::rational<double> obj_F(-12);
    //cout << "6) Test Equality(==) operator" << endl;
    //cout << "   Must be true: " << (obj_D==obj_F) << endl;
    //cout << "   Must be false: " << (obj_F==obj_C) << endl << endl;

    //cout << "7) Test Inequality(!=) operator" << endl;
    //cout << "   Must be true: " << (obj_C!=obj_F) << endl;
    //cout << "   Must be false: " << (obj_F!=obj_D) << endl << endl;

    //cout << "8) Test Less than(<) operator" << endl;
    //cout << "   Must be true: " << (obj_F<obj_C) << endl;
    //cout << "   Must be false: " << (obj_E<obj_F) << endl;
    //cout << "   Must be false: " << (obj_F<obj_D) << endl << endl;

    //cout << "9) Test Greater than(>) operator" << endl;
    //cout << "   Must be false: " << (obj_F>obj_C) << endl;
    //cout << "   Must be true: " << (obj_E>obj_F) << endl;
    //cout << "   Must be false: " << (obj_F>obj_D) << endl << endl;

```

```
62      //cout << "10) Test Less than or equals to(<=) operator" << endl;
63      //cout << "    Must be true: " << (obj_F<=obj_C) << endl;
64      //cout << "    Must be false: " << (obj_E<=obj_F) << endl;
65      //cout << "    Must be true: " << (obj_F<=obj_D) << endl << endl;
66
67      //cout << "11) Test Greater than or equals to(>=) operator" << endl;
68      //cout << "    Must be false: " << (obj_F>=obj_C) << endl;
69      //cout << "    Must be true: " << (obj_E>=obj_F) << endl;
70      //cout << "    Must be true: " << (obj_F>=obj_D) << endl << endl;
71
72      //cout << "12) Test Reduced form and negative denominator" << endl;
73      //cout << "    obj_C(31488,117): " << obj_C.numerator() << ", " << obj_C.den
    ominator() << endl;
74      //cout << "    obj_D(48,-4): " << obj_D.numerator() << ", " << obj_D.denomin
    ator() << endl << endl;
75
76      ra::math::rational<double> obj_G(-9,0);
77      //cout << "13) Test Condition when denominator is zero" << endl;
78      //cout << "    obj_G(-9,0): " << obj_G.numerator() << ", " << obj_G.denomina
    tor() << endl << endl;
79
80      //cout << "14) Test Prefix Increment(++obj) and Decrement(--obj) operators"
    << endl;
81      //cout << "    obj_D: " << obj_D.numerator() << ", " << obj_D.denominator()
    << endl;
82      //cout << "    increment: " << (++obj_D).numerator() << ", " << obj_D.denomi
    nator() << endl;
83      //cout << "    decrement: " << (--obj_D).numerator() << ", " << obj_D.denomi
    nator() << endl << endl;
84
85      //cout << "15) Test Postfix Increment(obj++) and Decrement(obj--) operators"
     << endl;
86      //cout << "    obj_D: " << obj_D.numerator() << ", " << obj_D.denominator()
    << endl;
87      //cout << "    increment: " << (obj_D++).numerator() << ", " << obj_D.denomi
    nator() << endl;
88      //cout << "    See change after: " << obj_D.numerator() << ", " << obj_D.den
    ominator() << endl;
89      //cout << "    decrement: " << (obj_D--).numerator() << ", " << obj_D.denomi
    nator() << endl;
90      //cout << "    See change after: " << obj_D.numerator() << ", " << obj_D.den
    ominator() << endl << endl;
91
92      ra::math::rational<double> obj_H(-9.776,1.33);
93      //cout << "16) Test case where a decimal points is used for the numerator an
    d denominator " << endl;
94      //cout << "    obj_H: " << obj_H.numerator() << ", " << obj_H.denominator()
    << endl << endl;
95
96      ra::math::rational<float> obj_I(8,10);
97      ra::math::rational<float> obj_J(1,5);
98      ra::math::rational<float> obj_K(2,3);
99      //cout << "17) Test Operator(+=) and (-=) and (*=) and (/=)" << endl;
100     //cout << "    (8/10) += (1/5): " << (obj_I+=obj_J).numerator() << "/" << ob
    j_I.denominator() << endl;
101     //cout << "    (prev ans) -= (2/3): " << (obj_I-=obj_K).numerator() << "/" <
    < obj_I.denominator() << endl;
102     //cout << "    (prev ans) *= (2/3): " << (obj_I*=obj_K).numerator() << "/" <
    < obj_I.denominator() << endl;
103     //cout << "    (prev ans) /= (1/5): " << (obj_I/=obj_J).numerator() << "/" <
    < obj_I.denominator() << endl << endl;
104
```

```
105     //cout << "18) Test Unary minus(-) and Unary plus(+)" << endl;
106     //cout << "    obj_H: " << obj_H.numerator() << ", " << obj_H.denominator()
    << endl;
107     //cout << "    Unary Plus: " << (+obj_H).numerator() << "/" << obj_H.denomin
    ator() << endl;
108     //cout << "    Unary Minus: " << (-obj_H).numerator() << "/" << obj_H.denomi
    nator() << endl;
109
110     //cout << "19) Test Binary operators (+), (-), (*), and (/)" << endl;
111     //cout <<"     "<<obj_I<<" + "<<obj_J<< " = " <<(obj_I+obj_J)<<endl;
112     //cout <<"     "<<obj_I<<" - "<<obj_J<< " = " <<(obj_I-obj_J)<<endl;
113     //cout <<"     "<<obj_I<<" * "<<obj_J<< " = " <<(obj_I*obj_J)<<endl;
114     //cout <<"     "<<obj_I<<" / "<<obj_J<< " = " <<(obj_I/obj_J)<<endl<<endl;
115
116     //cout << "20) Test Stream extractor" << endl;
117     //std::cin >> obj_K;
118     //cout << obj_K << endl;
119
120
121
122
123     return 0;
124 }
```