

SWE3 Übung 04

1 INHALTSVERZEICHNIS

2	Beispiel 1: Rational erweitert	3
2.1	Lösungsidee	3
2.1.1	Konstruktor ohne Parameter	3
2.1.2	Konstruktor mit einer Zahl als Parameter	3
2.1.3	Konstruktor mit 2 Zahlen als Parameter	3
2.1.4	Konstruktor mit einem rational_t object als Parameter	3
2.1.5	Get_numerator.....	3
2.1.6	Get_denominator	4
2.1.7	Is_negative.....	4
2.1.8	Is_positive.....	4
2.1.9	Is_zero	4
2.1.10	Print	4
2.1.11	Scan	4
2.1.12	As_string.....	4
2.1.13	Is_consistent.....	4
2.1.14	Normalize	4
2.1.15	Gcd.....	4
2.1.16	To_positive	4
2.1.17	Operator=	4
2.1.18	Operator==	4
2.1.19	Operator!=.....	4
2.1.20	Operator<	4
2.1.21	Operator>	4
2.1.22	Operator<=	5
2.1.23	Operator>=	5
2.1.24	Operator+=	5
2.1.25	Operator-=.....	5
2.1.26	Operator*=	5
2.1.27	Operator/=.....	5
2.1.28	Operator+	5
2.1.29	Operator-.....	5
2.1.30	Operator*	5

2.1.31	Operator/.....	5
2.1.32	Operatoren +,-,*,/ mit rational_t und int.....	5
2.1.33	Operator<<	5
2.1.34	Operator>>	5
2.1.35	Konstruktor ohne Parameter	6
2.1.36	Konstruktor mit einer Zahl als Parameter	6
2.1.37	Konstruktor mit einem number_t object als Parameter	6
2.1.38	Get_number	6
2.1.39	Add	6
2.1.40	Sub	6
2.1.41	Mul.....	6
2.1.42	Div.....	6
2.2	Code.....	6
2.3	Testfälle	7

Zeitaufwand: 10h

2 BEISPIEL 1: RATIONAL ERWEITERT

2.1 LÖSUNGSIDEE

Ziel dieser Übung war, die Klasse `rational_t` so zu verändern/zu erweitern, dass Bruchzahlen sowohl von ganzen als auch von rationalen Zahlen gebildet werden können. Zusätzlich sollte eine Klasse `number_t` implementiert werden, welche auch ganze oder rationale Zahlen beinhalten kann. Die Klasse `rational_t` kann auch vom Typ `number_t` sein.

Anforderungen an den Datentyp T:

- +
- -
- *
- /
- +=
- -=
- *=
- /=
- ==
- !=
- <
- >
- <=
- >=
- - (negate)
- `std::cout << t`

für die arithmetischen Operatoren und die Vergleichsoperatoren gilt die Anwendung auf zwei Objekte vom Typ T.

Klasse `rational_t<T>`:

2.1.1 Konstruktor ohne Parameter

Der Zähler wird mit dem zero-Element und der Nenner wird mit dem one-Element belegt.

2.1.2 Konstruktor mit einer Zahl als Parameter

Der Zähler wird mit dem Parameter und der Nenner wird mit dem one-Element belegt.

2.1.3 Konstruktor mit 2 Zahlen als Parameter

Zähler und Nenner werden mit den 2 übergebenen Zahlen belegt. Falls der Parameter für den Nenner dem zero-Element entspricht (Division durch 0) dann wird eine Exception geworfen.

2.1.4 Konstruktor mit einem `rational_t` object als Parameter

Zähler und Nenner werden mit Zähler und Nenner des übergebenen Objekts belegt.

2.1.5 `Get_numerator`

Der Zähler wird zurückgeliefert.

2.1.6 Get_denominator

Der Nenner wird zurückgeliefert.

2.1.7 Is_negative

Gibt True zurück, wenn die Bruchzahl negativ ist und False wenn sie positiv ist.

2.1.8 Is_positive

Macht das Gegenteil von is_negative

2.1.9 Is_zero

Gibt True zurück, wenn die Bruchzahl 0 ist (wenn der Zähler dem zero-Element entspricht) und False wenn sie nicht 0 ist.

2.1.10 Print

Gibt die Bruchzahl auf einem Output-Stream als String aus.

2.1.11 Scan

Liest die Bruchzahl von einem Input-Stream. Wirft eine Exception, falls ungültige Werte eingegeben werden.

2.1.12 As_string

Liefert die Bruchzahl als String zurück.

2.1.13 Is_consistent

Prüft, ob der Nenner dem zero-Element entspricht. Entspricht er nicht dem zero-Element, wird True zurückgeliefert, ansonsten False.

2.1.14 Normalize

Konvertiert die Bruchzahl in ihre kanonischen Repräsentanten. Dafür wird der größte gemeinsame Teiler von Zähler und Nenner ermittelt und die beiden werden durch diesen dividiert. Falls sowohl Zähler als auch Nenner negativ sind, werden die Minus weggelassen.

2.1.15 Gcd

Ermittelt den größten gemeinsamen Teiler zweier Zahlen.

2.1.16 To_positive

Wandelt eine Zahl in ihr positives Gegenstück um, falls diese negativ ist.

2.1.17 Operator=

Überlädt den Zuweisungsoperator und erlaubt damit eine rationale Zahl nur mithilfe eines „=“ mit einer anderen rationalen Zahl zu initialisieren.

2.1.18 Operator==

Zwei Bruchzahlen können über „==“ auf Gleichheit überprüft werden.

2.1.19 Operator!=

Zwei Bruchzahlen können über „!=“ auf Ungleichheit überprüft werden.

2.1.20 Operator<

Über „<“ kann überprüft werden, ob eine Bruchzahl kleiner ist als die andere.

2.1.21 Operator>

Über „>“ kann überprüft werden, ob eine Bruchzahl größer ist als die andere.

2.1.22 Operator<=

Über „<=“ kann überprüft werden, ob eine Bruchzahl kleiner oder gleich ist als die andere.

2.1.23 Operator>=

Über „>=“ kann überprüft werden, ob eine Bruchzahl größer oder gleich ist als die andere.

2.1.24 Operator+=

Zum Addieren zweier Brüche werden folgende Formeln angewandt:

Zähler = Zähler1 * Nenner2 + Zähler2 * Nenner1

Nenner = Nenner1 * Nenner2;

2.1.25 Operator-=

Beim Subtrahieren zweier Brüche, wird das Vorzeichen des Zählers der rechten Zahl umgekehrt und anschließend werden die Brüche addiert.

$$2/3 - 7/8 = 2/3 + (-7/8)$$

2.1.26 Operator*=

Falls einer der beiden Brüche 0 ist, wird auch das Ergebnis zu 0 (0/1). Anderenfalls werden sowohl die beiden Zähler als auch die beiden Nenner multipliziert.

2.1.27 Operator/=

Fall die rechte Zahl den Wert 0 hat, wird eine Exception geworfen. Ansonsten wird folgende Formel angewendet:

Zähler = Zähler1 * Nenner2

Nenner = Nenner1 * Zähler2

2.1.28 Operator+

Der += Operator wird aufgerufen.

2.1.29 Operator-

Der -= Operator wird aufgerufen.

2.1.30 Operator*

Der *= Operator wird aufgerufen.

2.1.31 Operator/

Der /= Operator wird aufgerufen.

2.1.32 Operatoren +, -, *, / mit rational_t und int

Diese Operatoren überladen die Addition, Subtraktion, Multiplikation und Division einer ganzen Zahl mit einer Bruchzahl. Hierbei wird aus der ganzen Zahl ein Bruch ($5 = 5/1$) und dann wird wie oben beschrieben gerechnet.

2.1.33 Operator<<

Print wird aufgerufen, der Output-Stream-Parameter wird mitübergeben.

2.1.34 Operator>>

Scan wird aufgerufen, der Input-Stream-Parameter wird mitübergeben.

Klasse number_t<T>

2.1.35 Konstruktor ohne Parameter

Die Zahl wird mit dem zero-Element des entsprechenden Datentyps belegt.

2.1.36 Konstruktor mit einer Zahl als Parameter

Die Zahl wird mit der übergebenen Zahl belegt.

2.1.37 Konstruktor mit einem number_t object als Parameter

Die Zahl wird mit der Zahl des übergebenen Objekts belegt.

2.1.38 Get_number

Der Wert der Zahl wird zurückgegeben.

2.1.39 Add

Addiert ein number_t Objekt zum aktuellen number_t Objekt.

2.1.40 Sub

Subtrahiert ein number_t Objekt vom aktuellen number_t Objekt.

2.1.41 Mul

Multipliziert ein number_t Objekt mit dem aktuellen number_t Objekt.

2.1.42 Div

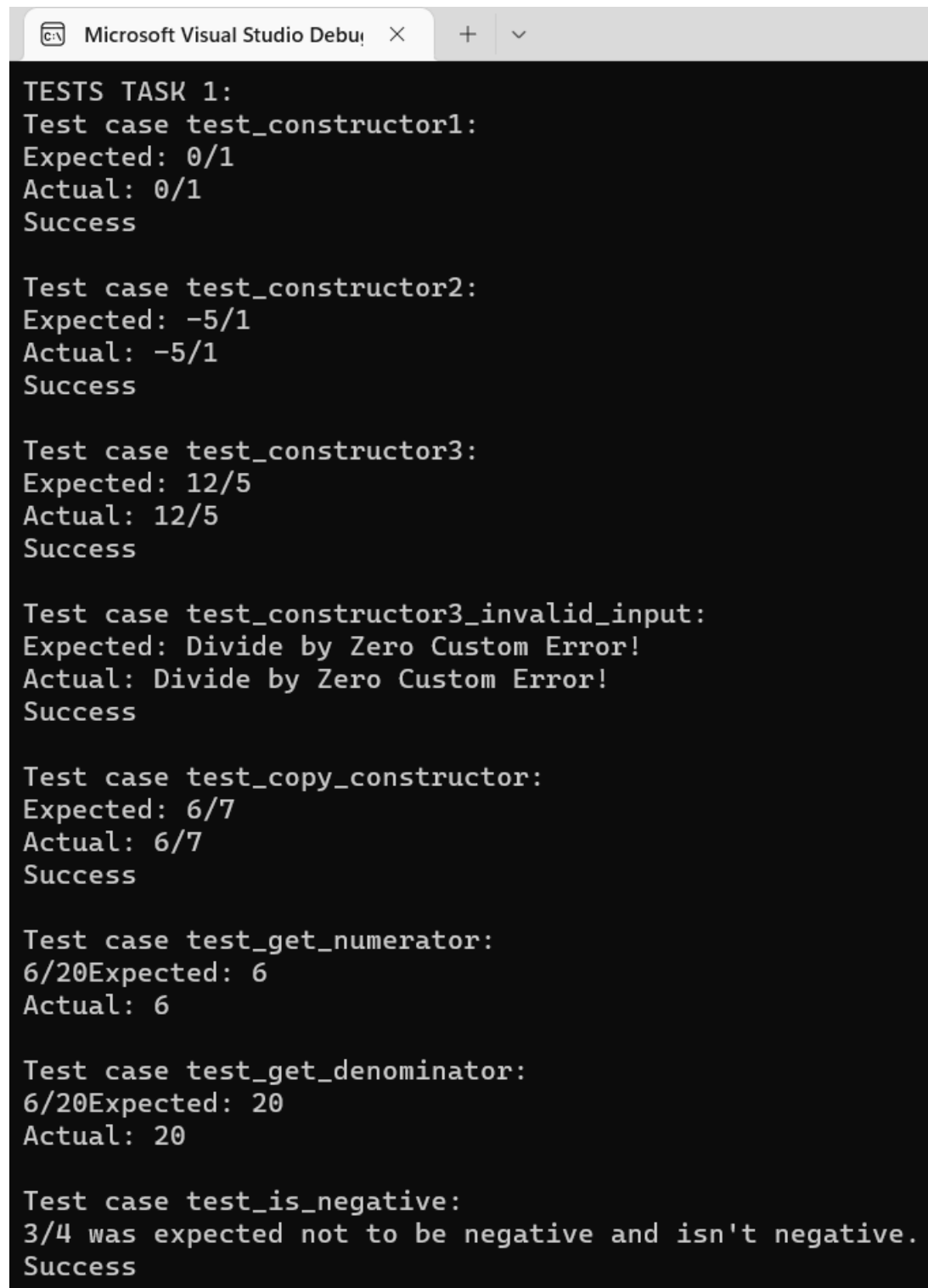
Dividiert das aktuelle number_t Objekt durch ein number_t Objekt.

Die überladenen Operatoren erfüllen ihre entsprechende Funktion.

2.2 CODE

siehe SWE3_Huspek_Ue04

2.3 TESTFÄLLE



```
Microsoft Visual Studio Debug Console
TESTS TASK 1:
Test case test_constructor1:
Expected: 0/1
Actual: 0/1
Success

Test case test_constructor2:
Expected: -5/1
Actual: -5/1
Success

Test case test_constructor3:
Expected: 12/5
Actual: 12/5
Success

Test case test_constructor3_invalid_input:
Expected: Divide by Zero Custom Error!
Actual: Divide by Zero Custom Error!
Success

Test case test_copy_constructor:
Expected: 6/7
Actual: 6/7
Success

Test case test_get_numerator:
6/20Expected: 6
Actual: 6

Test case test_get_denominator:
6/20Expected: 20
Actual: 20

Test case test_is_negative:
3/4 was expected not to be negative and isn't negative.
Success
```

```
Microsoft Visual Studio Debug Console
Test case test_is_positive:
-3/4 was expected not to be positive and isn't positive.
Success

Test case test_is_positive:
3/-4 was expected not to be positive and isn't positive.
Success

Test case test_is_positive:
-3/-4 was expected to be positive and is positive.
Success

Test case test_is_zero:
3/4 was expected not to be zero and isn't zero.
Success

Test case test_is_zero:
0/5 was expected to be zero and is zero.
Success

Test case test_constructor1_double:
Expected: 0.0/1.0
Actual: 0/1
Success

Test case test_constructor2_double:
Expected: -5.4/1.0
Actual: -5.4/1
Success

Test case test_constructor3_double:
Expected: 12.1/5.3
Actual: 12.1/5.3
Success

Test case test_constructor3_invalid_input_double:
Expected: Divide by Zero Custom Error!
Actual: Divide by Zero Custom Error!
Success
```



```
Microsoft Visual Studio Debug Console
Test case test_copy_constructor_double:
Expected: 6.8/7.3
Actual: 6.8/7.3
Success

Test case test_get_numerator_double:
6.1/20.5
Expected: 6.1
Actual: 6.1
Success

Test case test_get_denominator_double:
6.1/20.5
Expected: 20.5
Actual: 20.5
Success

Test case test_is_negative:
3.5/4.34 was expected not to be negative and isn't negative.
Success

Test case test_is_negative:
-3.23/4.779 was expected to be negative and is negative.
Success

Test case test_is_negative:
3.87/-4.99 was expected to be negative and is negative.
Success

Test case test_is_negative:
-3.22/-4.1 was expected not to be negative and isn't negative.
Success

Test case test_is_positive:
3.5/4.34 was expected to be positive and is positive.
Success
```

```
Microsoft Visual Studio Debug Console
Test case test_is_positive:
-3.23/4.779 was expected not to be positive and isn't positive.
Success

Test case test_is_positive:
3.87/-4.99 was expected not to be positive and isn't positive.
Success

Test case test_is_positive:
-3.22/-4.1 was expected to be positive and is positive.
Success

Test case test_is_zero:
3.5/4.34 was expected not to be zero and isn't zero.
Success

Test case test_is_zero:
0/5.6 was expected to be zero and is zero.
Success

TESTS TASK 2:
Test case test_constructor1_number_t:
Expected: 0/1
Actual: 0/1
Success

Test case test_constructor2_number_t:
Expected: -5/1
Actual: -5/1
Success

Test case test_constructor3_number_t:
Expected: 12/5
Actual: 12/5
Success
```

```
Microsoft Visual Studio Debug Console
Test case test_constructor3_invalid_input_number_t:
Expected: Divide by Zero Custom Error!
Actual: Divide by Zero Custom Error!
Success

Test case test_copy_constructor_number_t:
Expected: 6/7
Actual: 6/7
Success

TEST TASK 4:
Test case test_get_numerator_number_t:
6/20
Expected: 6
Actual: 6
Success

Test case test_get_denominator_number_t:
6/20
Expected: 20
Actual: 20
Success

Test case test_add_number_t:
4/5 + -2/3 =
Expected: 2/15
Actual: 2/15
Success

2/3 + -2/3 =
Expected: 0/1
Actual: 0/1
Success

-2/3 + 0/3 =
Expected: -2/3
Actual: -2/3
Success
```

```
Microsoft Visual Studio Debug Console
0/3 + -2/3 =
Expected: -2/3
Actual: -2/3
Success

5 + 4/5 =
Expected: 29/5
Actual: 29/5
Success

Test case test_sub_number_t:
4/5 - -2/3 =
Expected: 22/15
Actual: 22/15
Success

2/3 - 2/3 =
Expected: 0/1
Actual: 0/1
Success

-2/3 - 0/3 =
Expected: -2/3
Actual: -2/3
Success

0/3 - 5/6 =
Expected: -5/6
Actual: -5/6
Success

5 - 4/5 =
Expected: 21/5
Actual: 21/5
Success
```

```
Microsoft Visual Studio Debug Console
Test case test_mul_number_t:
12/4 * 3/2 =
Expected: 9/2
Actual: 9/2
Success

12/4 * 0/1 =
Expected: 0/1
Actual: 0/1
Success

0/1 * 12/4 =
Expected: 0/1
Actual: 0/1
Success

7/-8 * -5/3 =
Expected: 35/24
Actual: 35/24
Success

13 * -5/3 =
Expected: -65/3
Actual: -65/3
Success

Test case test_div_number_t:
12/4 / 3/2 =
Expected: 2/1
Actual: 2/1
Success

12/4 / 0/1 =
Expected: Divide by Zero Custom Error!
Actual: Divide by Zero Custom Error!
Success
```

```
Microsoft Visual Studio Debug Console
0/1 / 12/4 =
Expected: 0/1
Actual: 0/1
Success

7/-8 / -5/3 =
Expected: 21/40
Actual: 21/40
Success

13 / -5/3 =
Expected: 39/-5
Actual: 39/-5
Success

Test case test_assignment_op_number_t:
Expected: 4/7
Actual: 4/7
Success

Test case test_equal_op_number_t:
19/7 == 19/7
Expected: 1
Actual: 1
Success

19/7 == 8/5
Expected: 0
Actual: 0
Success

Test case test_not_equal_op_number_t:
19/7 != 19/7
Expected: 0
Actual: 0
Success
```

```
Microsoft Visual Studio Debug Console
19/7 != 8/5
Expected: 1
Actual: 1
Success

Test case test_greater_op_number_t:
19/7 > 22/7
Expected: 0
Actual: 0
Success

19/7 > 8/5
Expected: 1
Actual: 1
Success

Test case test_greater_or_equal_op_number_t:
19/7 >= 19/7
Expected: 1
Actual: 1
Success

19/7 >= 8/5
Expected: 1
Actual: 1
Success

19/7 >= 8/2
Expected: 0
Actual: 0
Success

Test case test_less_op_number_t:
19/7 < 22/7
Expected: 1
Actual: 1
Success
```

```
Microsoft Visual Studio Debug Console
19/7 < 8/5
Expected: 0
Actual: 0
Success

Test case test_less_or_equal_op_number_t:
19/7 <= 19/7
Expected: 1
Actual: 1
Success

19/7 <= 8/5
Expected: 0
Actual: 0
Success

19/7 <= 8/2
Expected: 1
Actual: 1
Success

Test case test_is_negative:
3/4 was expected not to be negative and isn't negative.
Success

Test case test_is_negative:
-3/4 was expected to be negative and is negative.
Success

Test case test_is_negative:
3/-4 was expected to be negative and is negative.
Success

Test case test_is_negative:
-3/-4 was expected not to be negative and isn't negative.
Success

Test case test_is_positive:
3/4 was expected to be positive and is positive.
Success
```



```
Microsoft Visual Studio Debug Console
Test case test_is_positive:
-3/4 was expected not to be positive and isn't positive.
Success

Test case test_is_positive:
3/-4 was expected not to be positive and isn't positive.
Success

Test case test_is_positive:
-3/-4 was expected to be positive and is positive.
Success

Test case test_is_zero:
3/4 was expected not to be zero and isn't zero.
Success

Test case test_is_zero:
0/5 was expected to be zero and is zero.
Success

Test case test_write_to_stream_number_t:
Expected: 24/6
Actual: 24/6
Success
Printed to print_rational_number_t.txt
Success

Test case test_read_from_stream_number_t:
Expected: 7/6
Actual: 7/6
Success

Expected: 4/7
Actual: 4/7
Success
```

```
Microsoft Visual Studio Debug Console
TESTS TASK 8:
Test case test_inverse:
3/-5
Expected: -5/3
Actual: -5/3
Success

Test case test_inverse_double:
3.4/-5.6
Expected: -5.6/3.4
Actual: -5.6/3.4

Test case test_inverse_number_t:
3/-5
Expected: -5/3
Actual: -5/3
Success

TESTS TASK 9:
Test case test_assignment_op:
Expected: 4/7
Actual: 4/7
Success

Test case test_assignment_op_double:
Expected: 4.5/7.2
Actual: 4.5/7.2
Success

Test case test_equal_op:
19/7 == 19/7
Expected: 1
Actual: 1
Success

19/7 == 8/5
Expected: 0
Actual: 0
Success
```

```
Microsoft Visual Studio Debug Console
Test case test_equal_op_double:
19.5/7.66 == 19.5/7.66
Expected: 1
Actual: 1
Success

19.5/7.66 == 19.4/7.88
Expected: 0
Actual: 0
Success

Test case test_not_equal_op:
19/7 != 19/7
Expected: 0
Actual: 0
Success

19/7 != 8/5
Expected: 1
Actual: 1
Success

Test case test_not_equal_op_double:
19.5/7.66 != 19.5/7.66
Expected: 0
Actual: 0
Success

19.5/7.66 != 19.4/7.88
Expected: 1
Actual: 1
Success

Test case test_greater_op:
19/7 > 22/7
Expected: 0
Actual: 0
Success
```

```
Microsoft Visual Studio Debug Console
19/7 > 8/5
Expected: 1
Actual: 1
Success

Test case test_greater_op_double:
19.5/7.2 > 22.5/7.2
Expected: 0
Actual: 0
Success

19.5/7.2 > 8.4/5.5
Expected: 1
Actual: 1
Success

Test case test_greater_or_equal_op:
19/7 >= 19/7
Expected: 1
Actual: 1
Success

19/7 >= 8/5
Expected: 1
Actual: 1
Success

19/7 >= 8/2
Expected: 0
Actual: 0
Success

Test case test_greater_or_equal_op_double:
19.44/7.9 >= 19.44/7.9
Expected: 1
Actual: 1
Success
```

```
Microsoft Visual Studio Debug Console
19.44/7.9 >= 8.32/5.4
Expected: 1
Actual: 1
Success

19.44/7.9 >= 8.32/2.2
Expected: 0
Actual: 0
Success

Test case test_less_op:
19/7 < 22/7
Expected: 1
Actual: 1
Success

19/7 < 8/5
Expected: 0
Actual: 0
Success

Test case test_less_op_double:
19.5/7.2 < 22.5/7.2
Expected: 1
Actual: 1
Success

19.5/7.2 < 8.4/5.5
Expected: 0
Actual: 0
Success

Test case test_less_or_equal_op:
19/7 <= 19/7
Expected: 1
Actual: 1
Success
```

```
Microsoft Visual Studio Debug Console
19/7 <= 8/5
Expected: 0
Actual: 0
Success

19/7 <= 8/2
Expected: 1
Actual: 1
Success

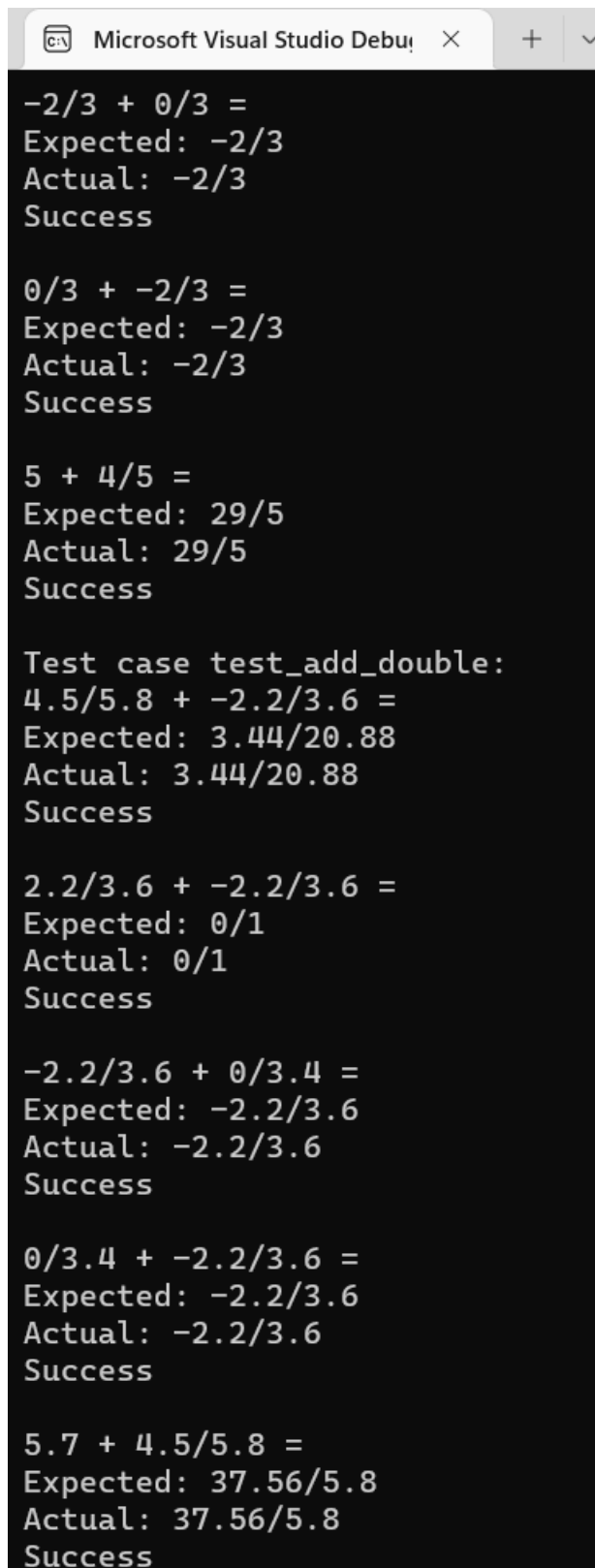
Test case test_less_or_equal_op_double:
19.44/7.9 <= 19.44/7.9
Expected: 1
Actual: 1
Success

19.44/7.9 <= 8.32/5.4
Expected: 0
Actual: 0
Success

19.44/7.9 <= 8.32/2.2
Expected: 1
Actual: 1
Success

Test case test_add:
4/5 + -2/3 =
Expected: 2/15
Actual: 2/15
Success

2/3 + -2/3 =
Expected: 0/1
Actual: 0/1
Success
```



```
Microsoft Visual Studio Debug Console
-2/3 + 0/3 =
Expected: -2/3
Actual: -2/3
Success

0/3 + -2/3 =
Expected: -2/3
Actual: -2/3
Success

5 + 4/5 =
Expected: 29/5
Actual: 29/5
Success

Test case test_add_double:
4.5/5.8 + -2.2/3.6 =
Expected: 3.44/20.88
Actual: 3.44/20.88
Success

2.2/3.6 + -2.2/3.6 =
Expected: 0/1
Actual: 0/1
Success

-2.2/3.6 + 0/3.4 =
Expected: -2.2/3.6
Actual: -2.2/3.6
Success

0/3.4 + -2.2/3.6 =
Expected: -2.2/3.6
Actual: -2.2/3.6
Success

5.7 + 4.5/5.8 =
Expected: 37.56/5.8
Actual: 37.56/5.8
Success
```

```
Microsoft Visual Studio Debug Console
Test case test_sub:
4/5 - -2/3 =
Expected: 22/15
Actual: 22/15
Success

2/3 - 2/3 =
Expected: 0/1
Actual: 0/1
Success

-2/3 - 0/3 =
Expected: -2/3
Actual: -2/3
Success

0/3 - 5/6 =
Expected: -5/6
Actual: -5/6
Success

5 - 4/5 =
Expected: 21/5
Actual: 21/5
Success

Test case test_sub_double:
4.5/5.8 - -2.2/3.6 =
Expected: 28.96/20.88
Actual: 28.96/20.88
Success

2.2/3.6 - 2.2/3.6 =
Expected: 0/1
Actual: 0/1
Success
```



```
Microsoft Visual Studio Debug Console
-2.2/3.6 - 0/3.4 =
Expected: -2.2/3.6
Actual: -2.2/3.6
Success

0/3.4 - 5.5/6.7 =
Expected: -5.5/6.7
Actual: -5.5/6.7
Success

5.4 - 4.5/5.8 =
Expected: 26.82/5.8
Actual: 26.82/5.8
Success

Test case test_mul:
12/4 * 3/2 =
Expected: 9/2
Actual: 9/2
Success

12/4 * 0/1 =
Expected: 0/1
Actual: 0/1
Success

0/1 * 12/4 =
Expected: 0/1
Actual: 0/1
Success

7/-8 * -5/3 =
Expected: 35/24
Actual: 35/24
Success

13 * -5/3 =
Expected: -65/3
Actual: -65/3
Success
```

```
Microsoft Visual Studio Debug Console
Test case test_mul_double:
12.5/4.3 * 3.8/2.48 =
Expected: 47.5/10.664
Actual: 47.5/10.664
Success

12.5/4.3 * 0/1.2 =
Expected: 0/1
Actual: 0/1
Success

0/1.2 * 12.5/4.3 =
Expected: 0/1
Actual: 0/1
Success

7.4/-8.6 * -5.3/3.78 =
Expected: 39.22/32.508
Actual: 39.22/32.508
Success

13.54 * -5.3/3.78 =
Expected: -71.762/3.78
Actual: -71.762/3.78
Success

Test case test_div:
12/4 / 3/2 =
Expected: 2/1
Actual: 2/1
Success

12/4 / 0/1 =
Expected: Divide by Zero Custom Error!
Actual: Divide by Zero Custom Error!
Success
```

```
Microsoft Visual Studio Debug Console
0/1 / 12/4 =
Expected: 0/1
Actual: 0/1
Success

7/-8 / -5/3 =
Expected: 21/40
Actual: 21/40
Success

13 / -5/3 =
Expected: 39/-5
Actual: 39/-5
Success

Test case test_div_double:
12.5/4.3 / 3.8/2.48 =
Expected: 31/16.34
Actual: 31/16.34
Success

12.5/4.3 / 0/1.2 =
Expected: Divide by Zero Custom Error!
Actual: Divide by Zero Custom Error!
Success

0/1.2 / 12.5/4.3 =
Expected: 0/1
Actual: 0/1
Success

7.4/-8.6 / -5.3/3.78 =
Expected: 27.972/45.58
Actual: 27.972/45.58
Success

13.54 / -5.3/3.78 =
Expected: 51.1812/-5.3
Actual: 51.1812/-5.3
Success
```

```
Microsoft Visual Studio Debug Console

Test case test_write_to_stream:
Expected: 24/6
Actual: 24/6
Success

Printed to print_rational.txt
Success

Test case test_write_to_stream_double:
Expected: 24.8/6.77
Actual: 24.8/6.77
Success
Printed to print_rational_double.txt
Success

Test case test_read_from_stream:
Expected: 45/3
Actual: 45/3
Success

Expected: 43/8
Actual: 43/8
Success

Test case test_read_from_stream_double:
Expected: 5.6/7.88
Actual: 5.6/7.88
Success

Expected: 2.5/7.8
Actual: 2.5/7.8
Success
```

```
print_rational - Editor

Datei  Bearbeiten  Ansicht

24/6
```

print_rational_double - Editor		
Datei	Bearbeiten	Ansicht
24.8/6.77		

print_rational_number_t - Editor		
Datei	Bearbeiten	Ansicht
24/6		