1. Input commands:

```
$ g++ -Wall -g operations.cpp -o test
$ gdb test
(gdb) run --arithmetic-operations encrypted_message.txt
test.txt
```

Symptom:

Aborted

Outputs:

```
Multidivide #1: 0 (expected 5).
```

Error messages:

```
assertion "multidivide(f,g,c,5,g) == 5" failed: file
"operations.cpp", line 70, function: int
arithmetic_operations()
Thread 1 "test" received signal SIGABRT, Aborted.
```

Expected:

The result of "`multidivide(f,g,c,5,g)`" should be the same as expected to fulfill the judgment of assert().

Debugger Info:

I ran the program in gdb.

```
(gdb) p e
$13 = 36
```

And:

```
(gdb) p g
$15 = 3
```

But in comments:

```
int e = b - 3*a + 5*c;      //  32
int g = e - (b/c) + d + 20; // -1
```

Analysis:

The codes themselves are correct while the expected result and the actual result are not the same, so the assert() function will mistakenly abort the program even as it's actually flawless.

Replacement:

Ln55, arithmetic_operations(), operations.cpp:

```
int e = b - 3*a + 4*c;      //  32
```

Ln57, arithmetic_operations(), operations.cpp:

```
int g = e - (b/c) + d + 24; // -1
```

Ln58, arithmetic_operations(), operations.cpp:

```
int h = (f/c) / a + 1;       //  3
```

Ln61, arithmetic_operations(), operations.cpp:

```
int p = (f / e) – h - 1;     // -1
```

Ln63, arithmetic_operations), operations.cpp:

```
int r = g + m + 2*p + n;     // -8
```

Overall environment:

System: Win10 64bits

Editor: Visual Studio 2017

Compiler: Cygwing

2. Input commands:

```
$ g++ -Wall -g operations.cpp -o test
$ gdb test
(gdb) run --arithmetic-operations encrypted_message.txt
test.txt
```

Symptom:

Syntax error.

Outputs:

```
Multidivide #5: 0 (expected 0.1).
```

Error messages:

None

Expected:

The result of multidivide() #5 should be 0.1

Debugger Info:

I ran the program in gdb.

```
(gdb) p s
$1 = 0
(gdb) p zeropointone
$2 = 0
```

But as expected, s and zeropointone should be 0.1

Analysis:

In c++, when an integer is divided by another integer, the result will be an integer, so in Ln64, arithmetic_operations(), operations.cpp:

```
float s = a / f;           //  0.1
```

s was assigned to 0 instead of 0.1.

And in Ln359, multidivide(), operations.cpp:

```
float f = (((((numerator) / d1) / d2) / d3) / d4);
```

f was assigned to an integer, too.

Replacement:

Ln64, arithmetic_operations(), operations.cpp:

```
float s = (float)a / f;     //  0.1
```

Ln359, multidivide(), operations.cpp:

```
float f = (((((float(numerator)) / d1) / d2) /
d3) / d4);
```

3. **Input commands:**

```
        $ g++ -Wall -g operations.cpp -o test
        $ gdb test
        (gdb) run --file-operations
encrypted_message.txt test.txt
```

Symptom:

Cerr

Outputs:

```
Usage: ./test operations infile outfile
Couldn't start operations.
```

Error messages:

None

Expected:

Should have read arguments successfully.

Debugger Info:

```
306      if(argc == 4) {
(gdb) s
307        std::cerr << "Usage: " << argv[0]
<< " operations infile outfile" <<
std::endl;
(gdb) p argc
$1 = 4
```

Analysis:

When argc == 4, there shall not be cerr, however, according to the cerr messages, the judgment should return true when there's invalid number of argc, in other words, when argc is not 4.

Replacement:

Ln306, file_operations(), operations.cpp:

```
if(argc != 4) {
```

4. **Input commands:**

```
        $ g++ -Wall -g operations.cpp -o test
        $ gdb test
        (gdb) run --file-operations
encrypted_message.txt test.txt
```

Symptom:

Aborted

Outputs:

```
Successfully opened the input file.
Successfully read in 0 bytes of data.
```

Error messages:

```
assertion "infile.gcount() == length" failed:
file "operations.cpp", line 342, function: bool
file_operations(int, char**, char*&, int&)
```

Expected:

The value of gcount() printed should not be 0 and should be same as length.

Debugger Info:

```
(gdb) p length
$1 = -1
```

Analysis:

The variable length should give out a position of infile(tellg), which should have been a positive number in normal situation. However, when the value of length is -1, it means that the ifstream infile hasn't been opened successfully. But the if statements to judge whether the file has been correctly read didn't pop out any cerrs, so I guess there must be something wrong with the judgment statements. I checked and found that the condition of the judgment was wrong. It worked in opposite way.

Replacement:

Ln318, file_operations(), operations.cpp:

```
if(!infile) {
```

5. **Input commands:**

```
$ g++ -Wall -g operations.cpp -o test
$ gdb test
(gdb) run --list-operations
encrypted_message.txt test.txt
```

Symptom:

Aborted

Outputs:

```
elderberry quart nectarine orange zwetschge
pomegranate durian grape yellow squash fig iodine
strawberry tangerine jujube lemon mango cherry
uglyfruit apple watermelon kiwi
-13449 letters did not ever appear in the fruit
names.
```

Error messages:

```
assertion "*l1.begin() == 'A'" failed: file
"operations.cpp", line 527, function: int
list_operations()
```

Expected:

The list should begin with "A".

Debugger Info:

None

Analysis:

The value assigned to l1.begin() has been wrong, which means the last value push backed into the list was wrong, so I checked the constructor of the list and found that the order of capitalized letters being assigned was wrong.

Replacement:

Ln431, list_operations(), operations.cpp:

```
for(char c =  'Z'; c >= 'A'; c--) {
```

6. **Input commands:**

```
$ g++ -Wall -g operations.cpp -o test
$ gdb test
(gdb) run --list-operations
encrypted_message.txt test.txt
```

Symptom:

None

Outputs:

```
elderberry quart nectarine orange zwetschge
pomegranate durian grape yellow squash fig iodine
strawberry tangerine jujube lemon mango cherry
uglyfruit apple watermelon kiwi
-13449 letters did not ever appear in the fruit
names.
List bugs are NOT FIXED
```

Error messages:

None

Expected:

Should not be "bugs are not fixed" and the count of letters should not be -13449.

Debugger Info:

```
(gdb) p count
$1 = -13449
```

Analysis:

The integer variable count had not been initialized, so the default value of integer would be assigned to the variable which is -13449.

Replacement:

Ln517, list_operations(), operations.cpp:

```
int count = 0;
```

7. **Input commands:**

```
$ g++ -Wall -g operations.cpp -o test
$ gdb test
(gdb) run --list-operations encrypted_message.txt test.txt
```

Symptom:

None

Outputs:

```
elderberry quart nectarine orange zwetschge
pomegranate durian grape yellow squash fig iodine
strawberry tangerine jujube lemon mango cherry
uglyfruit apple watermelon kiwi
0 letters did not ever appear in the fruit names.
List bugs are NOT FIXED
```

Error messages:

None

Expected:

Should not be "bugs are not fixed" and the count of letters should not be 0.

Debugger Info:

```
(gdb) p count
$1 = 0
```

Analysis:

The count hadn't been ever changed since it was initialized, so the for loop including the judgment statements must be wrong. And I checked to find that an unexpected break was commanded inside the if statement. If an element in the list didn't fit the judgment, the program should go on for the next element but not directly break the loop.

Replacement:

Ln520, list_operations(), operations.cpp:

```
continue;
```

8. **Input commands:**

```
$ g++ -Wall -g operations.cpp -o test
$ gdb test
(gdb) run --list-operations encrypted_message.txt test.txt
```

Symptom:

None

Outputs:

```
elderberry quart nectarine orange zwetschge
pomegranate durian grape yellow squash fig iodine
strawberry tangerine jujube lemon mango cherry
uglyfruit apple watermelon kiwi
2 letters did not ever appear in the fruit names.
List bugs are NOT FIXED
```

Error messages:

None

Expected:

Should not be "bugs are not fixed" and the list of fruits is different from the expected output.

Debugger Info:

```
(gdb) p count
$1 = 0
```

Analysis:

The code ++i is different with i++ where i++ means the program will read the value of the variable first then, add one to it. However, in the original codes, ++fruit_itr was telling the program to add one to the pointer first then read the value, which is apparently wrong.

Replacement:

Ln497, list_operations(), operations.cpp:

```
fruits.erase(fruit_itr++);
```