CSCI-1200 Data Structures — Homework 11

Ziniu Yu RIN# 661540383

Compile original codes.

```
operations.cpp:130:27: warning: variable 'length' is uninitialized when used here [-Wuninitialized]
  char* buffer = new char[length];
                          ^~~~~~
operations.cpp:127:13: note: initialize the variable 'length' to silence this warning
  int length;
            ^
             = 0
operations.cpp:182:1: warning: control may reach end of non-void function [-Wreturn-type]
}
^
operations.cpp:171:43: warning: variable 'placeholder' is uninitialized when used here [-Wuninitialized]
  float fracpart = modf(sqrt(sumsquares), placeholder);
                                          ^~~~~~~~~~~
operations.cpp:166:22: note: initialize the variable 'placeholder' to silence this warning
  double* placeholder; // will store the integer part from modf
                     ^
                      = NULL
operations.cpp:242:7: warning: variable 'sum' is uninitialized when used here [-Wuninitialized]
      sum += array[x][y];
      ^~~
operations.cpp:239:10: note: initialize the variable 'sum' to silence this warning
  int sum;
         ^
          = 0
operations.cpp:367:36: warning: expression result unused [-Wunused-value]
  for(uint i = 0; i < all.size(); i+1) {
                                  ~^~
operations.cpp:378:27: warning: comparison of unsigned expression >= 0 is always true [-Wtautological-compare]
  for(uint i=counter-1; i >= 0; --i) {
                        ~ ^  ~
operations.cpp:314:7: warning: variable 'counter' is uninitialized when used here [-Wuninitialized]
      counter++;
      ^~~~~~~
operations.cpp:310:14: note: initialize the variable 'counter' to silence this warning
  int counter;
             ^
              = 0
```

1. operations.cpp: line 127: change to "**int length = 0;**"
   Notice that the following line "char* buffer = new char[length];" create an array of length 0 which is meaningless, it may in the wrong location.

2. operations.cpp: Move line 130 "**char* buffer = new char[length];**" to where we have read the length.

3. operations.cpp: Add "**return -1;**" in the end of pythagoras function.

4. operations.cpp: line 166: change to "**double* placeholder = 0;**"

5. operations.cpp: line 242: change to "**int sum = 0;**"

6. operations.cpp: line 367: change to "**for(uint i = 0; i < all.size(); i++) {**"

7. operations.cpp: line 310: change to "**int counter = 0;**"

8. operations.cpp: line 378: change to "**for(uint i=counter-1; i > 0; --i) {**"

Save and recompile. Run the program.

```
Multidivide #1: 0 (expected 5).
Assertion failed: (multidivide(f,g,c,5,g) == 5), function arithmetic_operations, file operations.cpp, line 65.
Abort trap: 6
```

9. 10. 11. 12. 13. operations.cpp: line 45 to line 59: force change some data type to float, and use the correct approximation.
Add std::cout statement after each number to make sure the numbers are correct.

```
  // set up some variables
  int a = 10;
  int b = 46;
  int c = 4;
  int d = c - b;                          // -42
  int e = b - 3 * a + 4 * c;              //  32
  int f = 2 * b + 2 * c;                  //  100
  int g = e - (b / c) + d + 20;           // -1
  int h = ceilf((float(f) / float(c)) / float(a)); //  3
  int m = (d / h) / 7;                    // -2
  int n = g + m;                          // -3
  int p = floorf((float(f) / float(e)) - h - 1);  // -1
  int q = f + 2 * d;                      // 16
  int r = g + m + p + n -1;               // -8
  float s = float(a) / float(f);          //  0.1
```

Save and recompile. Run the program.

```
Multidivide #1: 5 (expected 5).
Multidivide #2: -10 (expected -10).
Multidivide #3: -2 (expected -2).
Multidivide #4: -5 (expected -5).
Multidivide #5: 0 (expected 0.1).
Assertion failed: (close_enough(zeropointone, s)), function arithmetic_operations, file operations.cpp, line 90.
Abort trap: 6
```

14. operations.cpp: notice that in Multidivide #5 the result is not we expect. So we should change the multidivide function to "**float multidivide(float numerator, float d1, float d2, float d3, float d4) {**"

Save and recompile. Run the program.

```
Multidivide #1: 5 (expected 5).
Multidivide #2: -10 (expected -10).
Multidivide #3: -2 (expected -2).
Multidivide #4: -5 (expected -5).
Multidivide #5: 0.1 (expected 0.1).
Finished the arithmetic operations
Arithmetic bugs are FIXED
Usage: ./11.out datafile
Couldn't start operations.
Assertion failed: (array[1][2] == -1), function array_operations, file operations.cpp, line 210.
Abort trap: 6
```

15. operations.cpp: line 110: it should be "**if(argc != 2) {**"

Save and recompile. Run the program.

```
Multidivide #1: 5 (expected 5).
Multidivide #2: -10 (expected -10).
Multidivide #3: -2 (expected -2).
Multidivide #4: -5 (expected -5).
Multidivide #5: 0.1 (expected 0.1).
Finished the arithmetic operations
Arithmetic bugs are FIXED
That file could not be opened!
```

16. operations.cpp: line 121 change to "**if(!infile) {**" It should give error message when something is wrong.

Save and recompile. Run the program.

```
Multidivide #1: 5 (expected 5).
Multidivide #2: -10 (expected -10).
Multidivide #3: -2 (expected -2).
Multidivide #4: -5 (expected -5).
Multidivide #5: 0.1 (expected 0.1).
Finished the arithmetic operations
Arithmetic bugs are FIXED
Successfully opened the input file.
Successfully read in 131 bytes of data.
Finished the file operations
Assertion failed: (array[1][2] == -1), function array_operations, file operations.cpp, line 211.
Abort trap: 6
```

17. 18. 19. 20. 21. 22. operations.cpp: Correct boundary conditions and increment correctly.
line 194 **for(int x=0; x<size; ++x) {**
line 196 **for(int y=0; y<size; ++y) {**
line 204 **for(int x=0; x<size; ++x) {**
line 205 **for(int y=0; y<size; ++y) {**
line 229 **for(int x = 0; x < size; ++x, ++tmp_ptr) {**
line 231 **for(int y = 0; y < size; ++y, ++tmp_ptr2) {**

23. 24. 25. 26. 27. 28. operations.cpp: Modify Pythagoras function. Use absolute value in subtraction to prevent negative root. Reference: http://www.cplusplus.com/reference/cmath/modf/
line 169 **double placeholder = 0;**
line 174 **float fracpart = modf(sqrt(sumsquares), &placeholder);**
line 175 line 176 **if(fracpart == 0) return (int) placeholder;**
line 179 **float diffsquares = abs(y*y - x*x);**
line 180 **fracpart = modf(sqrt(diffsquares), &placeholder);**
line 181 line 182 **if(fracpart == 0) return (int) placeholder;**

Save and recompile. Run the program.

```
Multidivide #1: 5 (expected 5).
Multidivide #2: -10 (expected -10).
Multidivide #3: -2 (expected -2).
Multidivide #4: -5 (expected -5).
Multidivide #5: 0.1 (expected 0.1).
Finished the arithmetic operations
Arithmetic bugs are FIXED
Successfully opened the input file.
Successfully read in 131 bytes of data.
Finished the file operations
Printing the Pythagorean numbers array.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Omission~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Finished the array operations
Array bugs are FIXED
Assertion failed: (v1sum == 175), function vector_operations, file operations.cpp, line 300.
Abort trap: 6
```

29. operations.cpp: line 287: The code does different thing than the comment tells to do. It should be “**for(uint i=1; i<=10; ++i) {**“

30. operations.cpp: Modify the vector_sum function. Make it do the right calculations. The vector should be called by reference.

```
int vector_sum(std::vector<int>& inVec) {
  for(uint i=1; i<=inVec.size(); ++i) {
    inVec[i] = inVec[i] + inVec[i-1];
  }
  return inVec[inVec.size()-1];
}
```

Save and recompile. Run the program.

```
Multidivide #1: 5 (expected 5).
Multidivide #2: -10 (expected -10).
Multidivide #3: -2 (expected -2).
Multidivide #4: -5 (expected -5).
Multidivide #5: 0.1 (expected 0.1).
Finished the arithmetic operations
Arithmetic bugs are FIXED
Successfully opened the input file.
Successfully read in 131 bytes of data.
Finished the file operations
Printing the Pythagorean numbers array.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Omission~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Finished the array operations
Array bugs are FIXED
Assertion failed: (vector_compare(v1, v4)), function vector_operations, file operations.cpp, line 350.
```

31. 32. operations.cpp: Modify the vector_compare function. Change it to correct comparison. Make the vectors are called by reference.
line 271 **bool vector_compare(const std::vector<int>& v1, const std::vector<int>& v2) {**
line 274 **if(v1[i] <= v2[i]) {**

33. operations.cpp: line 373: Change it to “**if(all[i] % 3 == 0) {**“

34. operations.cpp: line 376: Change it to “**threes.push_back(all[i]);**”

35. operations.cpp: line 383: Change it to “**for(uint i=counter-1; i >= 0; --i) {**“

36. operations.cpp: line 370: Reset the counter to prevent segfault. Add “**counter = 0;**”

Save and recompile. Run the program.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Omission~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Finished the arithmetic operations
Arithmetic bugs are FIXED
Successfully opened the input file.
Successfully read in 131 bytes of data.
Finished the file operations
Printing the Pythagorean numbers array.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Omission~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Finished the array operations
Array bugs are FIXED
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Omission~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Finished the vector operations
Vector bugs are FIXED
Assertion failed: (t2.get_type() == "Isosceles"), function triangle_operations, file operations.cpp, line 411.
Abort trap: 6
```

37. triangle.h: line 26: Change to “**x1(x1_), y1(y1_), x2(x2_), y2(y2_), x3(x3_), y3(y3_), name(name_) {}**”

38. triangle.cpp: line 57: Correct the typo **return "Isosceles";**

Go through the triangle.cpp, correct some obvious bugs.

39. 40. 41. 42. triangle.cpp: get_area() function

```
float Triangle::get_area() const {
  double a = length_between(x1, y1, x2, y2);
  double b = length_between(x2, y2, x3, y3);
  double c = length_between(x1, y1, x3, y3);
  double s = (a+b+c)/2;
  return sqrt(s*(s-a)*(s-b)*(s-c));
}
```

43. triangle.cpp: get_perimeter() function

```
float Triangle::get_perimeter() const {
  return length_between(x1, y1, x2, y2) + length_between(x2, y2, x3, y3) +
length_between(x1, y1, x3, y3);
}
```

44. triangle.cpp: get_type() function, add more definitions of isosceles triangle

else if(close_enough(a,c) || close_enough(a,b) || close_enough(b,c)) {

Save and recompile. Run the program.

```
Decryption successful - good job!
```

Result:

```
A bright young coder named Lee
Wished to loop while i was 3
But when writing the =
He forgot its sequel
And thus looped infinitely
```