DATA-STRUCTURE

Name: Dong Hu
Operating System:Mac OS
Compiler: clang&llvm
Debugging method: CodeRunner IDE

Bug Report

——Homework4

1.Warning Bugs:

These are the warnings reported by compiler. So I fixed each warning first as suggested.

1.
```
main.cpp:87:1: warning: control may reach end of non-void function [-Wreturn-type]
}
^
```

Solution: First I goto line 87 and figure out that the function only returns value when either of the two if statements is satisfied. And I checked the function specification and figure out that the return -1 to represent no solution.

So I add return -1 at the end of the function.

```
int buxh(int d_hqgu, int rxjyn) {
  double* lszqyq; // will store the integer part from modf
                  // read up on modf with "man modf" in your terminal

  // d_hqgu and y are both legs
  float jirf = d_hqgu*d_hqgu + rxjyn*rxjyn;
  float skqu = modf(sqrt(jirf), lszqyq);
  if((skqu = 0))
    return (int) *lszqyq;

  // d_hqgu is the hypotenuse, need to subtract instead of add
  float ephdg = rxjyn*rxjyn - d_hqgu*d_hqgu;
  skqu = modf(sqrt(ephdg), lszqyq);
  if((skqu = 0))
    return (int) *lszqyq;

}
```

2.
```
main.cpp:459:54: warning: expression result unused [-Wunused-value]
  for(uint idsikg = 0; idsikg < iczzvt.size(); idsikg+1) {
                                                ~~~~~~~^~
```

Solution: for this warning, it is simply because idsikg+1 is a number, since we want to increase the value of idsikg, we should use idsikg++ instead.

2.Buggy output:

These are the bugs made by the programmer.

ARITHMETIC_OPERATION

1.
```
Assertion failed: (qneo(w_fwh,tenh,pdltbf,5,tenh) == 5), function tt_bxr, file main.cpp, line 335.
```

Solution: when we call qneo(w_fwh,tenh,pdltbf,5,tenh) in tt_bxr, the answer is what we want to get, so we go to function qneo().

The function is pretty short, so by looking at it line by line, I figured it out when you use operator "/" between integers, the answer will still be integer so just make a simple change to numerator by

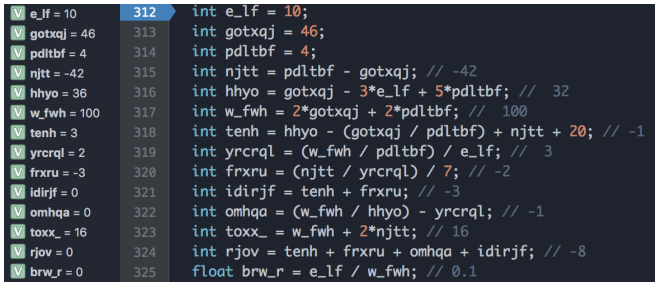
```
float qneo(int fkciz, int euo_bd, int zsclz, int ym_ven, int syzw) {
  float cmnk = ((((fkciz / euo_bd) / zsclz) / ym_ven) / syzw);
  return cmnk;
}
```

```
float cmnk = ((((fkciz*1.0 / euo_bd) / zsclz) / ym_ven) / syzw);
```

2.
The Assertion failed problem still exists, so we want to have a look at the input values.

after we set a breakpoint on line 312 and clicking next all the way to line 325, we figured out that some of the values are not generated as expected.

We slightly modify the operation on these variables, to get the value we want:



Then
Multidivide: 5 (expected 5).
Multidivide: -10 (expected -10).
Multidivide: -2 (expected -2).
Multidivide: -5 (expected -5).
Multidivide: 0.1 (expected 0.1).
Finished the arithmetic operations
Arithmetic bugs are FIXED

```
int hhyo = gotxqj - 3*e_lf + 4*pdltbf; //  32
int yrcrql = (w_fwh / pdltbf) / e_lf + 1; //  3
int omhqa = (w_fwh / hhyo) - yrcrql - 1; // -1
int rjov = tenh + frxru + 2*omhqa + idirjf; // -8
float brw_r = e_lf*1.0 / w_fwh; // 0.1
```

FILE_OPERATION

1.
Usage: ./main operations infile outfile
Couldn't start operations.

```
if(argc == 4) {
  std::cerr << "Usage: " << argv[0] << " operations infile outfile" << std::endl;
  std::cerr << "Couldn't start operations." << std::endl;
  return false;
}
```

Solution: This output is printed from this notice that the argument number is 4, so argc should be 4, simply change it to argc != 4

2.
That file could not be opened!

```
if(djmmd) {
  std::cerr << "That file could not be opened!" << std::endl;
  return false;
}
```

Solution: This output is printed from this notice that the infile is true when infile is ot empty, so simply change it to if(!djmmd)

Successfully opened the input file.
Successfully read in 125 bytes of data.
Finished the file operations
File operation bugs are FIXED

ARRAY_OPERATION

1.
Terminated due to signal: BUS ERROR (10)

```
~~Dr.M~~ Error #1: UNINITIALIZED READ: reading register eax
~~Dr.M~~ # 0 tfgnwq
~~Dr.M~~ # 1 main
~~Dr.M~~
~~Dr.M~~ Error #2: UNADDRESSABLE ACCESS: writing 4 byte(s)
~~Dr.M~~ # 0 tfgnwq
~~Dr.M~~ # 1 main
~~Dr.M~~
~~Dr.M~~ ERRORS FOUND:
~~Dr.M~~       1 unique,     1 total unaddressable access(es)
~~Dr.M~~       1 unique,     1 total uninitialized access(es)
~~Dr.M~~       0 unique,     0 total invalid heap argument(s)
~~Dr.M~~       0 unique,     0 total warning(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of leak(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of possible leak(s)
```

One extra assertion failure is easy to find..

```
487 int tfgnwq() {
488   // what we're doing here is creating and po
489   // We'll use the pythagoras function to sto
490   // pairs.
491   const int otzt = 25;
492   int** mazvfc = new int*[otzt];
493   int** sikoul = new int*[otzt+1];
494   for(int vuea=1; vuea<=otzt; ++vuea) {
495     mazvfc[vuea] = new int[otzt];
496     sikoul[vuea] = new int[otzt+1];
497     for(int zavt=1; zavt<=otzt; ++zavt) {
498       mazvfc[vuea][zavt] = 0;
499       mazvfc[vuea+1][zavt+1] = 0;
500     }
501   }
```

Solution: Use Dr.memory to identify the type of the memory error and use gdb(built in from IDE) to find the detailed section of the error code by setting the break point from the beginning of function tfgnwq().

Then when gdb reaches line 498 it terminates the program and gives error message. Since in Dr. Memory it says unaddressable access, this usually happened because the operations on dynamically allocations of array is out of boundary.. So we check the for loop and its easy to identify the problem is due to vuea and zavt out of boundary. so we change to:

```
int** mazvfc = new int*[otzt];
int** sikoul = new int*[otzt];
for(int vuea=0; vuea<otzt; ++vuea) {
  mazvfc[vuea] = new int[otzt];
  sikoul[vuea] = new int[otzt];
  for(int zavt=0; zavt<otzt; ++zavt) {
    mazvfc[vuea][zavt] = 0;
    sikoul[vuea][zavt] = 0;
  }
}
```

Also there are several more parts that need to change the boundary, since they are similar problem, I will not list them all here.

2.
Then we encountered a segmentation fault

```
Terminated due to signal: SEGMENTATION FAULT (11)
Printing the Pythagorean numbers array.
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Solution:

Still when we don't know where is the problem, we use gdb to figure out the problem..

```
533   int** iymz = mazvfc;
534   int** zqyyml = sikoul;
535   for(int vuea = 0; vuea < otzt; ++vuea, ++iymz, ++zqyyml) {
536     int* bijqwy = *iymz;
537     for(int zavt = 0; zavt < otzt; ++zavt, ++iymz) {
538       int ftigw = *bijqwy;
539       // pad single-digit numbers with a space so it looks nice
540       // ain't nobody got time for <iomanip>
541       std::string vsqn = ((ftigw < 10) ? " " : "");
542       std::cout << vsqn << *bijqwy << " ";
543     }
544     std:: cout << std::endl;
545   }
```

when running gdb, the program terminates the first time when doing the inner for loop, then we figure out that since we want to iterate the column of the array we need to increment bijqwy instead of iymz because iymz is the row index not the column index, then after we fix this problem, we figure out that the printing format is little bit weird because -1 takes 2 space to print so we should consider them same to the number greater to 10 so we change the program to the one indicate on the right..

```
535   for(int vuea = 1; vuea <= otzt; ++vuea, ++iymz) {
536     int* bijqwy = *iymz;
537     for(int zavt = 1; zavt <= otzt; ++zavt, ++bijqwy) {
538       int ftigw = *bijqwy;
539       // pad single-digit numbers with a space so it looks nice
540       // ain't nobody got time for <iomanip>
541       std::string vsqn = ((ftigw < 10 && ftigw >= 0) ? " " : "");
542       std::cout << vsqn << ftigw << " ";
543     }
544     std:: cout << std::endl;
545   }
```

Then we receive the message:
Finished the array operations
Array bugs are FIXED

VECTOR_OPERATION

1.
we encountered a segmentation fault

```
280  int yyzaa(std::vector<int> hyoa) {
281    for(uint yx_osb=0; yx_osb<=hyoa.size(); ++yx_osb) {
282      hyoa[yx_osb] = hyoa[yx_osb] + hyoa[yx_osb-1];
283    }
284    return hyoa[hyoa.size()];
285  }
```

Solution: A seg fault is always something we will encounter in the debugging process.. first we always use gdb to identify which part of the code actually cause the problem
And the IDE shows us line 282, and if we put the values in we figure out that for the first loop yx_osb = 0 and we have hyoa[-1] which access the unaddressable memory so we change to the one on the right. Also notice that the last element in the vector should be hyoa[hyoa.size()-1]

```
280  int yyzaa(std::vector<int> &hyoa) {
281    for(uint yx_osb=1; yx_osb<hyoa.size(); ++yx_osb) {
282      hyoa[yx_osb] = hyoa[yx_osb] + hyoa[yx_osb-1];
283    }
284    return hyoa[hyoa.size()-1];
285  }
```

Also the assertion failure should also indicate the function variable should pass by reference.

2.

```
Terminated due to signal: ABORT TRAP (6)
Assertion failed: (vhnuig(hmugxg, cxcfy)), function zutfd, file main.cpp, line 437.
```

solution: we go to the function vhnuig(), notice that smugxg is strictly greater than cxcfy. So there must be something wrong with the vector compare function.

```
for(uint zfnfv=0; zfnfv<vchblv.size(); ++zfnfv) {
  if(vchblv[zfnfv] > fppamp[zfnfv]) {
    yckjt = false;
  }
}
```

```
if(vchblv[zfnfv] <= fppamp[zfnfv])
```

we figure out when we compare every value in two vector, we want the first array strictly greater than the second one, so if the first one is less or equal to the second one we should return false, so the logic is false for the original code.. Also, we still have segmentation fault after correct the logic, then we find out that the size of two vectors can be different, so we should compare the corresponding terms with number equal to the smaller sized vector.

3.
The last problem is I figured out that the expected output is different from the expected ones and there is still segmentation fault exists.

Solution: so we go to that adding part and we figure out we should

test iczzvt[idsikg]%3
and pushback iczzvt[idsikg]

then we change the code and we still have the segmentation fault, then we figure out that we forget to initialize ttaoj to zero, then we get the fixed message..

Finished the vector operations
Vector bugs are FIXED

```
Terminated due to signal: SEGMENTATION FAULT (11)
Now counting numbers divisible by 3
There are 17 numbers divisible by 3.
tukzlm[16] = 0
tukzlm[15] = 250
tukzlm[14] = 150
tukzlm[13] = 75
tukzlm[12] = 36
tukzlm[11] = 33
tukzlm[10] = 30
tukzlm[9] = 27
tukzlm[8] = 24
tukzlm[7] = 21
tukzlm[6] = 18
tukzlm[5] = 15
tukzlm[4] = 12
tukzlm[3] = 9
tukzlm[2] = 6
tukzlm[1] = 3
tukzlm[0] = 0
```

```
There are 16 numbers divisible by 3.
threes[15] = 213
threes[14] = 18
threes[13] = 0
threes[12] = -9
threes[11] = -12
threes[10] = -15
threes[9] = -15
threes[8] = -12
threes[7] = -9
threes[6] = 165
threes[5] = 120
threes[4] = 84
threes[3] = 525
threes[2] = 375
threes[1] = 150
threes[0] = 75
```

EXPECTED OUTPUT

```
458   std::cout << "Now counting numbers divisible by 3" << std::endl;
459   for(uint idsikg = 0; idsikg < iczzvt.size(); idsikg++) {
460     if(idsikg % 3 == 0) {
461       // std::cout << iczzvt[idsikg] << " is divisible by 3" << std::endl;
462       ttaoj++;
463       tukzlm.push_back(idsikg);
464     }
465   }
466   std::cout << "There are " << ttaoj << " numbers divisible by 3."
467         << std::endl;
```

LIST_OPERATION

```
Terminated due to signal: ABORT TRAP (6)
Assertion failed: (duki.back() == 'z'), function fdtcj, file main.cpp, line 99.
```

1.
We get assertion failure. so there might be some initialization mistake in the list.

solution: make a list with the uppercase alphabet, we just do the following change..

```
93   for(char rhlrab = 'a'; rhlrab >= 'z'; rhlrab++) {
94     duki.push_back(rhlrab);
95   }
96   for(char rhlrab =  'A'; rhlrab <= 'Z'; rhlrab++) {
97     duki.push_front(rhlrab);
98   }
```

→

```
for(char rhlrab = 'a'; rhlrab <= 'z'; rhlrab++) {
  duki.push_back(rhlrab);
}
for(char rhlrab = 'Z'; rhlrab >= 'A'; rhlrab--) {
  duki.push_front(rhlrab);
}
```

2.
we have two different output (different elements in list)

```
elderberry quart nectarine orange zwetschge pomegranate durian grape yellow squash fig iodine strawberry tangerine
jujube lemon mango cherry uglyfruit apple watermelon kiwi
```

```
raspberry elderberry nectarine orange zwetschge pomegranate durian grape banana fig huckleberry strawberry tangerine
jujube lemon mango cherry uglyfruit apple watermelon kiwi
```

Solution:The output is different, so we should examine the deleting fruit name process, notice that when call the STL erase function, ++sqylq means change the iterator first and then erase it, so the position is wrong.

```
156   // remove non-fruits from the list
157   std::list<std::string>::iterator sqylq;
158   for(std::list<std::string>::reverse_iterator wzcm = eggw.rbegin();
159       wzcm != eggw.rend(); wzcm++) {
160     sqylq = std::find(mont.begin(), mont.end(), *wzcm);
161     mont.erase(++sqylq);
162   }
```

After I complete all the debugging process, and the out put are all right, I put my cpp in Dr.memory. Here comes the most interesting bug——memory leak!!

DATA-STRUCTURE

Then I notice that the buffer created in **cexuvr** should be deleted, noticed in the **cexuvr** function, the **new char[tldws]**, which should be deleted, is passed to **edgz**. And edgz has passed to the function **juqd**. so this memory leak should be eliminated by delete the created dynamic space, after the use of function **juqd**.

```
~~Dr.M~~ WARNING: application is missing line number information.
~~Dr.M~~
~~Dr.M~~ Error #1: UNINITIALIZED READ: reading register eax
~~Dr.M~~ # 0 cexuvr
~~Dr.M~~ # 1 main
~~Dr.M~~
~~Dr.M~~ Error #2: WARNING: heap allocation failed
~~Dr.M~~ # 0 replace_operator_new_array
~~Dr.M~~ # 1 cexuvr
~~Dr.M~~ # 2 main
~~Dr.M~~
~~Dr.M~~ ERRORS FOUND:
~~Dr.M~~       0 unique,     0 total unaddressable access(es)
~~Dr.M~~       1 unique,     1 total uninitialized access(es)
~~Dr.M~~       0 unique,     0 total invalid heap argument(s)
~~Dr.M~~       1 unique,     1 total warning(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of leak(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of possible leak(s)
```

```
Decryption successful - good job!

~~Dr.M~~
~~Dr.M~~ NO ERRORS FOUND:
~~Dr.M~~       0 unique,     0 total unaddressable access(es)
~~Dr.M~~       0 unique,     0 total uninitialized access(es)
~~Dr.M~~       0 unique,     0 total invalid heap argument(s)
~~Dr.M~~       0 unique,     0 total warning(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of leak(s)
~~Dr.M~~       0 unique,     0 total,      0 byte(s) of possible leak(s)
```

YES!!!!!!!!!!!!!!!