LAB [1], [01/10/2019] **MCS 253P**

Name: Yu Qin

Problem Description

Read numbers from a file, To get the average value of those numbers.
!: No dynamic allocation.

Additional Problem Specifics

Input file type: file.dat (and I later treat the file without the suffix, but no binary file.)
Number value overflow? Valid number?: Yes, number
Integer: No, floating numbers
Negative: Yes, maybe
Output format: numbers scale, avg number
Empty? : Given No empty, and I later give information for users.
too many input numbers? It has not too much differences between one input number and one million inputer number, since the process is designed to scan the file line by line.
too larger input numbers? Double is much enough; if really more larger, it is another trivial point.

Sample Input

1st) 1, 3 ------- 2
2nd) 10,20,30 ------- 20
3rd) -1 ----- - 1
4th) empty file
5th) different file name and suffix (eg., file.dat, file.txt, etc.)

Proposed Algorithm

Description:

The overall thought is to calculate the sum and the number count, and get the quotient
So, maintain a double for the sum, and a double for the count.
All the data should be read from the file.

Correctness:

It is based on the math concept. It works for both positive numbers and negative numbers.

Time Complexity:

It scans the file while update the sum and count, so it's O(n).

Space Complexity:

It just need a double number for sum and count, so it's O(1).

C++ Implementation of Algorithm

```
#include <iostream>
#include <fstream>
using namespace std;

int main(int argc, char** argv){
        //open file
        if (argc < 2){
                cout<<"You should input the file name.\n\n";
                return -0;
        }
        ifstream infile(argv[1]);
        if ( !infile.is_open() ){
                cout<<"Cannot open the file.\n\n";
                return -0;
        }

        //read file
        double int_temp;
        double sum = 0, count = 0, average = 0;
        while ( infile >> int_temp) {
                sum = sum + int_temp;
                count += 1;
        }
        infile.close();

        //calculate
        if (count == 0 ){
                cout<<"Your input file is empty.\n\n";
                return -0;
        } else{
                average = sum/count;
                cout<<"The average of the "<< count <<" numbers in file '" << argv[1] << "' is "<< average <<
endl<<endl;
                return 0;
        }
}
```

Edge Test Cases

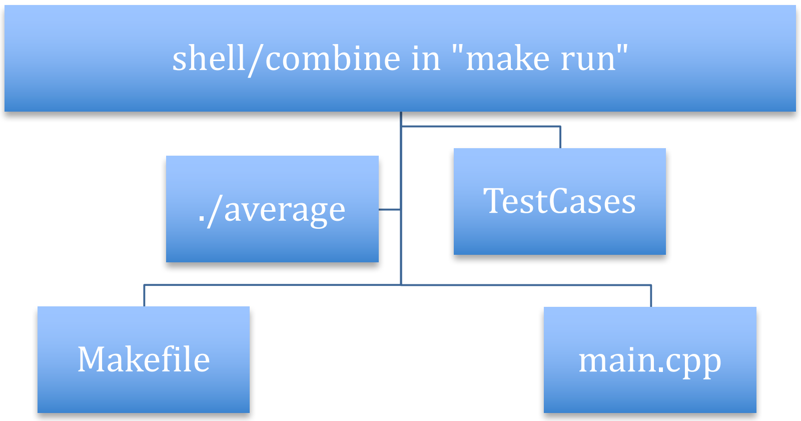
- sample description of an edge test case
    - output we expect (want)
    - output our algorithm produces
- no file name input in command
  expected: remind of "You should input the file name" and exit.
  output our algorithm produces: remind of "You should input the file name" and exit.
  conclusion: right 🆗
- file open fail (maybe due to wrong file name, lack of permission, etc.)
  expected: remind of "Cannot open the file." and exit.
  output our algorithm produces: remind of "Cannot open the file." and exit.
  conclusion: right 🆗
- empty file
  expected: remind of "Your input file is empty." and exit.
  output our algorithm produces: remind of "Your input file is empty." and exit.
  conclusion: right 🆗
- corner case of data are considered in former "Additional Problem Specifics" part.

Comments

Good question, basic and realistic to be used.
Be cautious to the input file processing.
Sometime we may need to handle the valid number input.

Solution Frame Description

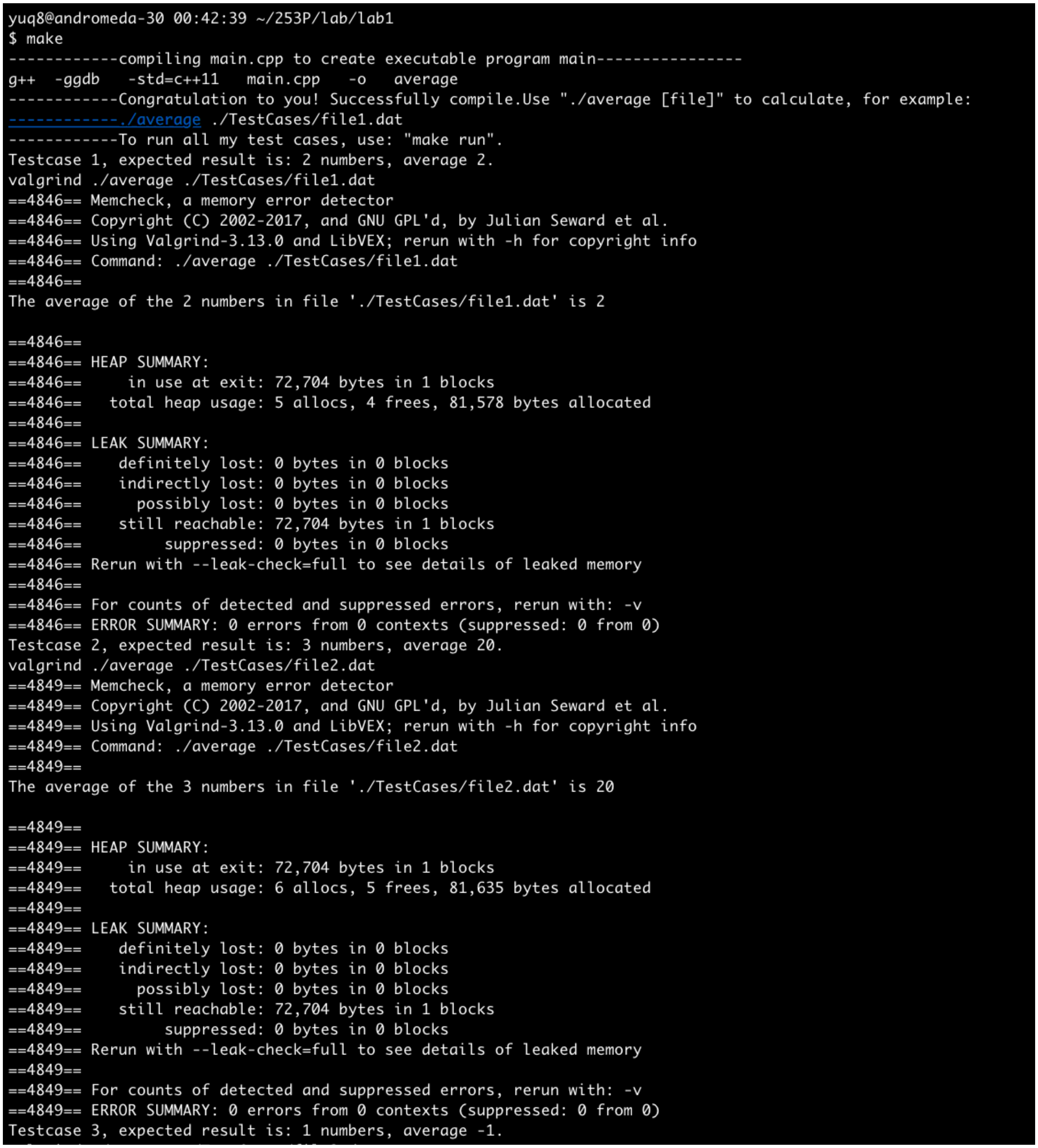
As required, “main_not_too_much_output_details.cpp” solve the solution.
But since updated requirement need to output all the detail inputs, the new “main.cpp” is to be created.
And to run all the test cases in one pass, a shell script is to facilitated to run the command.



But actually, since in new piazza says we can show the correctness by screenshot of several times, so, no shell is needed. So now, the “main.cpp” is just the former “main_not_too_much_output_details.cpp”.
And we just write all the test cases command in a Makefile, thus can builds and runs all my program just by “make run” (“make” can still support manually manipulate.)

Results Screenshots

========================start of the write-up=========================
Testcase 1, 2:

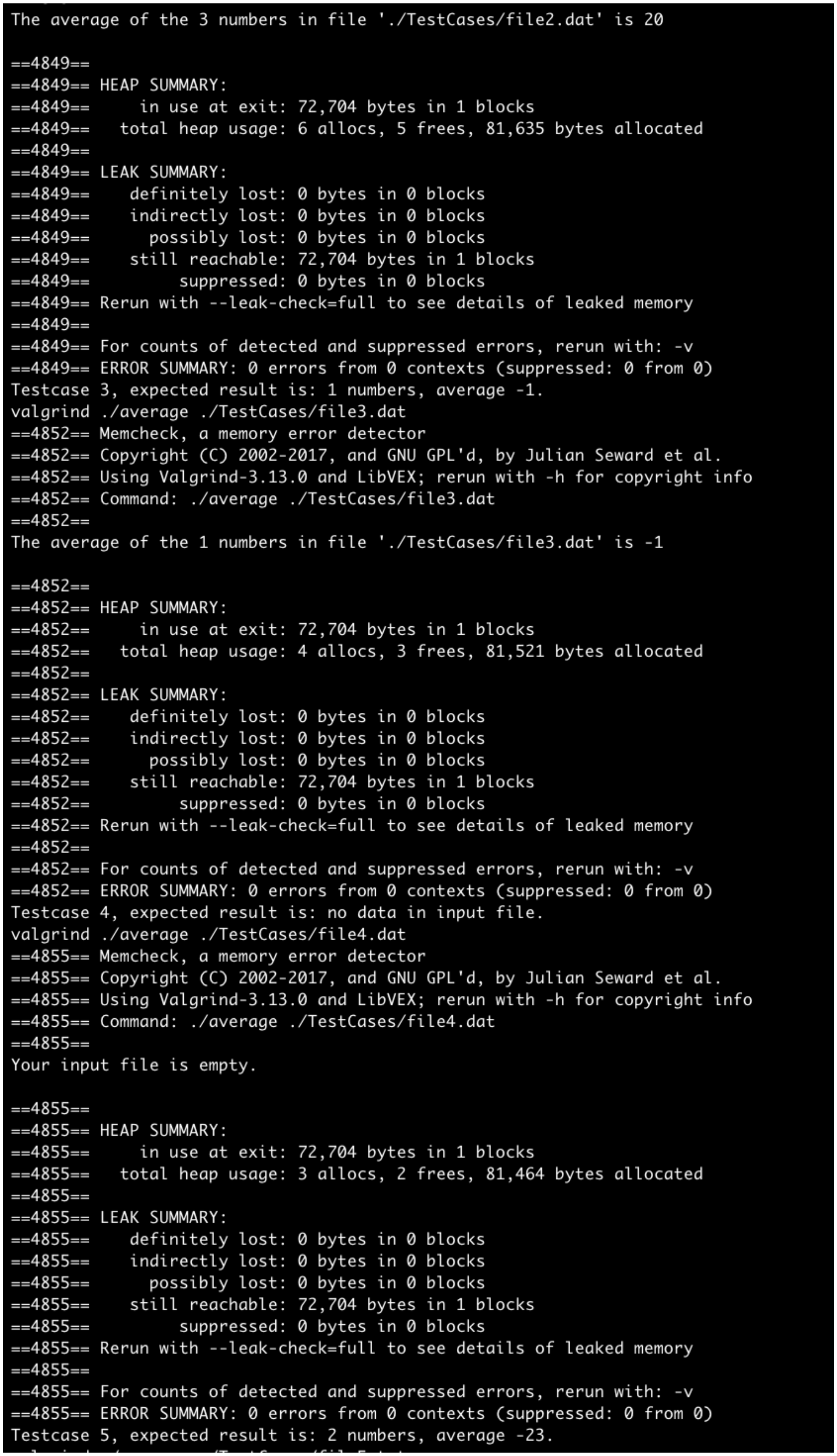


```
yuq8@andromeda-30 00:42:39 ~/253P/lab/lab1
$ make
------------compiling main.cpp to create executable program main----------------
g++  -ggdb   -std=c++11   main.cpp   -o   average
------------Congratulation to you! Successfully compile.Use "./average [file]" to calculate, for example:
------------./average ./TestCases/file1.dat
------------To run all my test cases, use: "make run".
Testcase 1, expected result is: 2 numbers, average 2.
valgrind ./average ./TestCases/file1.dat
==4846== Memcheck, a memory error detector
==4846== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4846== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4846== Command: ./average ./TestCases/file1.dat
==4846==
The average of the 2 numbers in file './TestCases/file1.dat' is 2

==4846==
==4846== HEAP SUMMARY:
==4846==     in use at exit: 72,704 bytes in 1 blocks
==4846==   total heap usage: 5 allocs, 4 frees, 81,578 bytes allocated
==4846==
==4846== LEAK SUMMARY:
==4846==    definitely lost: 0 bytes in 0 blocks
==4846==    indirectly lost: 0 bytes in 0 blocks
==4846==      possibly lost: 0 bytes in 0 blocks
==4846==    still reachable: 72,704 bytes in 1 blocks
==4846==         suppressed: 0 bytes in 0 blocks
==4846== Rerun with --leak-check=full to see details of leaked memory
==4846==
==4846== For counts of detected and suppressed errors, rerun with: -v
==4846== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
Testcase 2, expected result is: 3 numbers, average 20.
valgrind ./average ./TestCases/file2.dat
==4849== Memcheck, a memory error detector
==4849== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4849== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4849== Command: ./average ./TestCases/file2.dat
==4849==
The average of the 3 numbers in file './TestCases/file2.dat' is 20

==4849==
==4849== HEAP SUMMARY:
==4849==     in use at exit: 72,704 bytes in 1 blocks
==4849==   total heap usage: 6 allocs, 5 frees, 81,635 bytes allocated
==4849==
==4849== LEAK SUMMARY:
==4849==    definitely lost: 0 bytes in 0 blocks
==4849==    indirectly lost: 0 bytes in 0 blocks
==4849==      possibly lost: 0 bytes in 0 blocks
==4849==    still reachable: 72,704 bytes in 1 blocks
==4849==         suppressed: 0 bytes in 0 blocks
==4849== Rerun with --leak-check=full to see details of leaked memory
==4849==
==4849== For counts of detected and suppressed errors, rerun with: -v
==4849== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
Testcase 3, expected result is: 1 numbers, average -1.
```

Testcase 3, 4:



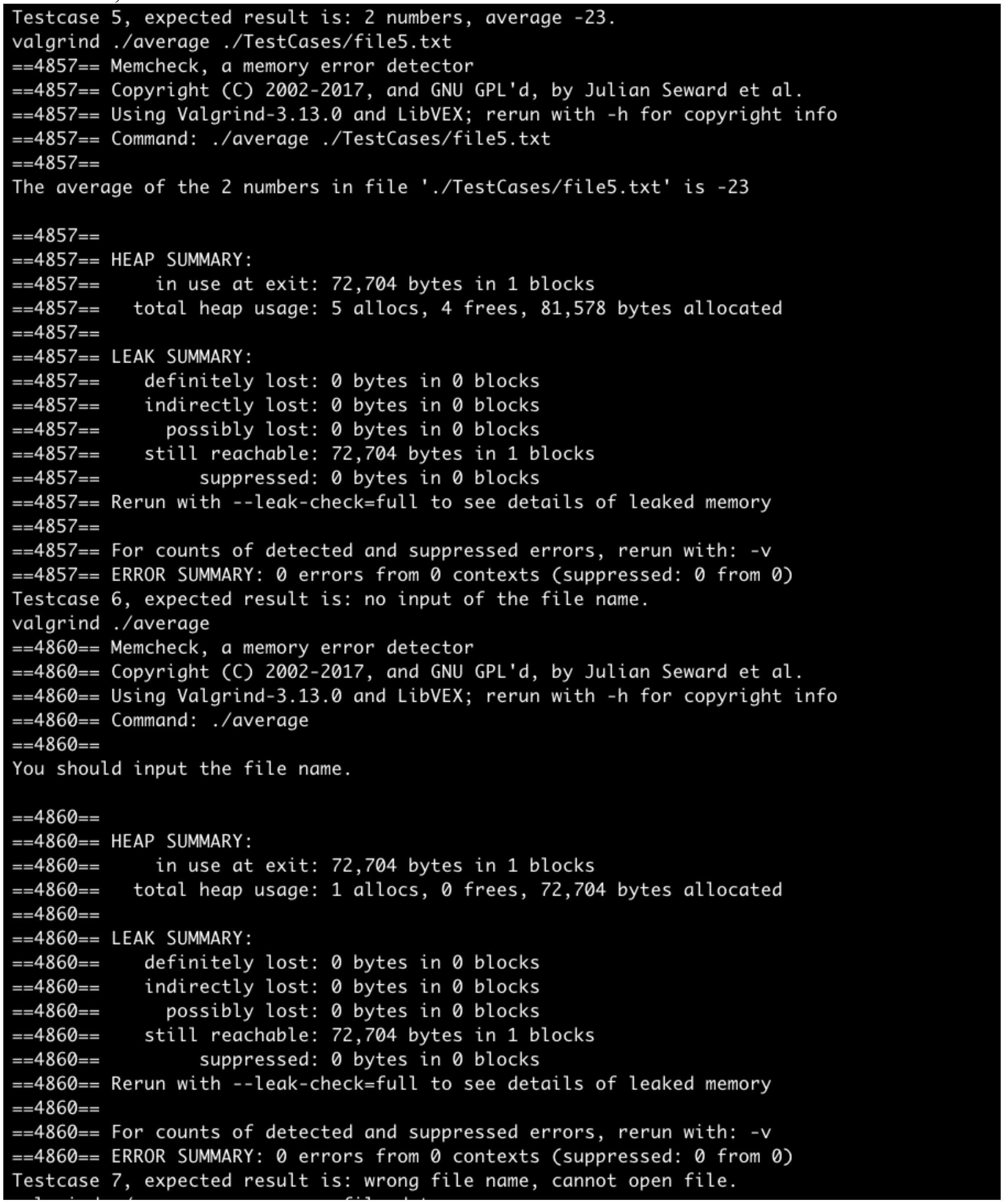
```
The average of the 3 numbers in file './TestCases/file2.dat' is 20

==4849==
==4849== HEAP SUMMARY:
==4849==     in use at exit: 72,704 bytes in 1 blocks
==4849==   total heap usage: 6 allocs, 5 frees, 81,635 bytes allocated
==4849==
==4849== LEAK SUMMARY:
==4849==    definitely lost: 0 bytes in 0 blocks
==4849==    indirectly lost: 0 bytes in 0 blocks
==4849==      possibly lost: 0 bytes in 0 blocks
==4849==    still reachable: 72,704 bytes in 1 blocks
==4849==         suppressed: 0 bytes in 0 blocks
==4849== Rerun with --leak-check=full to see details of leaked memory
==4849==
==4849== For counts of detected and suppressed errors, rerun with: -v
==4849== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
Testcase 3, expected result is: 1 numbers, average -1.
valgrind ./average ./TestCases/file3.dat
==4852== Memcheck, a memory error detector
==4852== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4852== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4852== Command: ./average ./TestCases/file3.dat
==4852==
The average of the 1 numbers in file './TestCases/file3.dat' is -1

==4852==
==4852== HEAP SUMMARY:
==4852==     in use at exit: 72,704 bytes in 1 blocks
==4852==   total heap usage: 4 allocs, 3 frees, 81,521 bytes allocated
==4852==
==4852== LEAK SUMMARY:
==4852==    definitely lost: 0 bytes in 0 blocks
==4852==    indirectly lost: 0 bytes in 0 blocks
==4852==      possibly lost: 0 bytes in 0 blocks
==4852==    still reachable: 72,704 bytes in 1 blocks
==4852==         suppressed: 0 bytes in 0 blocks
==4852== Rerun with --leak-check=full to see details of leaked memory
==4852==
==4852== For counts of detected and suppressed errors, rerun with: -v
==4852== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
Testcase 4, expected result is: no data in input file.
valgrind ./average ./TestCases/file4.dat
==4855== Memcheck, a memory error detector
==4855== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4855== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4855== Command: ./average ./TestCases/file4.dat
==4855==
Your input file is empty.

==4855==
==4855== HEAP SUMMARY:
==4855==     in use at exit: 72,704 bytes in 1 blocks
==4855==   total heap usage: 3 allocs, 2 frees, 81,464 bytes allocated
==4855==
==4855== LEAK SUMMARY:
==4855==    definitely lost: 0 bytes in 0 blocks
==4855==    indirectly lost: 0 bytes in 0 blocks
==4855==      possibly lost: 0 bytes in 0 blocks
==4855==    still reachable: 72,704 bytes in 1 blocks
==4855==         suppressed: 0 bytes in 0 blocks
==4855== Rerun with --leak-check=full to see details of leaked memory
==4855==
==4855== For counts of detected and suppressed errors, rerun with: -v
==4855== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
Testcase 5, expected result is: 2 numbers, average -23.
```

Testcase 5, 6:

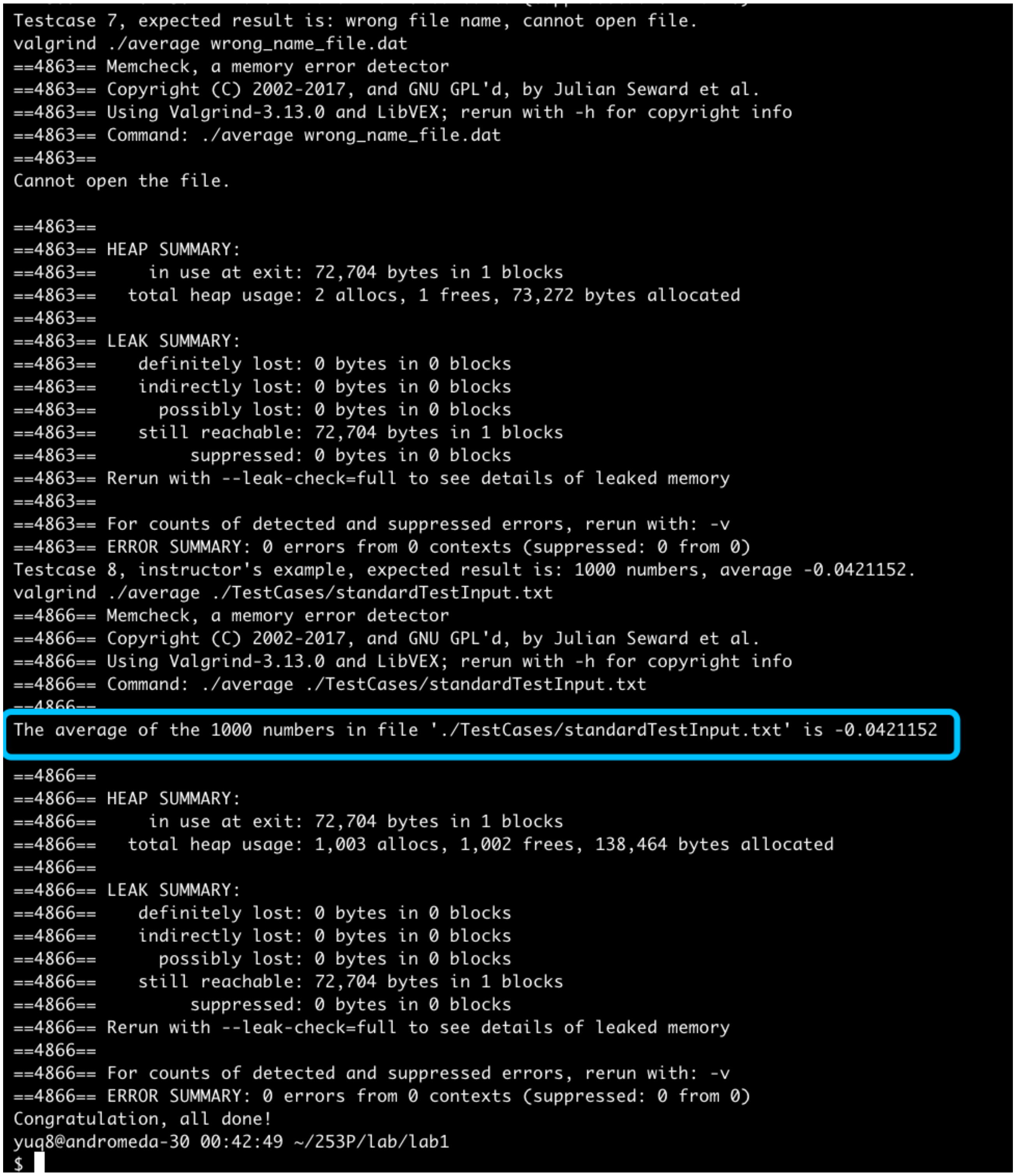


```
Testcase 5, expected result is: 2 numbers, average -23.
valgrind ./average ./TestCases/file5.txt
==4857== Memcheck, a memory error detector
==4857== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4857== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4857== Command: ./average ./TestCases/file5.txt
==4857==
The average of the 2 numbers in file './TestCases/file5.txt' is -23

==4857==
==4857== HEAP SUMMARY:
==4857==     in use at exit: 72,704 bytes in 1 blocks
==4857==   total heap usage: 5 allocs, 4 frees, 81,578 bytes allocated
==4857==
==4857== LEAK SUMMARY:
==4857==    definitely lost: 0 bytes in 0 blocks
==4857==    indirectly lost: 0 bytes in 0 blocks
==4857==      possibly lost: 0 bytes in 0 blocks
==4857==    still reachable: 72,704 bytes in 1 blocks
==4857==         suppressed: 0 bytes in 0 blocks
==4857== Rerun with --leak-check=full to see details of leaked memory
==4857==
==4857== For counts of detected and suppressed errors, rerun with: -v
==4857== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
Testcase 6, expected result is: no input of the file name.
valgrind ./average
==4860== Memcheck, a memory error detector
==4860== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4860== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4860== Command: ./average
==4860==
You should input the file name.

==4860==
==4860== HEAP SUMMARY:
==4860==     in use at exit: 72,704 bytes in 1 blocks
==4860==   total heap usage: 1 allocs, 0 frees, 72,704 bytes allocated
==4860==
==4860== LEAK SUMMARY:
==4860==    definitely lost: 0 bytes in 0 blocks
==4860==    indirectly lost: 0 bytes in 0 blocks
==4860==      possibly lost: 0 bytes in 0 blocks
==4860==    still reachable: 72,704 bytes in 1 blocks
==4860==         suppressed: 0 bytes in 0 blocks
==4860== Rerun with --leak-check=full to see details of leaked memory
==4860==
==4860== For counts of detected and suppressed errors, rerun with: -v
==4860== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
Testcase 7, expected result is: wrong file name, cannot open file.
```

Testcase 7, 8:



```
Testcase 7, expected result is: wrong file name, cannot open file.
valgrind ./average wrong_name_file.dat
==4863== Memcheck, a memory error detector
==4863== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4863== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4863== Command: ./average wrong_name_file.dat
==4863==
Cannot open the file.

==4863==
==4863== HEAP SUMMARY:
==4863==     in use at exit: 72,704 bytes in 1 blocks
==4863==   total heap usage: 2 allocs, 1 frees, 73,272 bytes allocated
==4863==
==4863== LEAK SUMMARY:
==4863==    definitely lost: 0 bytes in 0 blocks
==4863==    indirectly lost: 0 bytes in 0 blocks
==4863==      possibly lost: 0 bytes in 0 blocks
==4863==    still reachable: 72,704 bytes in 1 blocks
==4863==         suppressed: 0 bytes in 0 blocks
==4863== Rerun with --leak-check=full to see details of leaked memory
==4863==
==4863== For counts of detected and suppressed errors, rerun with: -v
==4863== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
Testcase 8, instructor's example, expected result is: 1000 numbers, average -0.0421152.
valgrind ./average ./TestCases/standardTestInput.txt
==4866== Memcheck, a memory error detector
==4866== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4866== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==4866== Command: ./average ./TestCases/standardTestInput.txt
==4866==
The average of the 1000 numbers in file './TestCases/standardTestInput.txt' is -0.0421152

==4866==
==4866== HEAP SUMMARY:
==4866==     in use at exit: 72,704 bytes in 1 blocks
==4866==   total heap usage: 1,003 allocs, 1,002 frees, 138,464 bytes allocated
==4866==
==4866== LEAK SUMMARY:
==4866==    definitely lost: 0 bytes in 0 blocks
==4866==    indirectly lost: 0 bytes in 0 blocks
==4866==      possibly lost: 0 bytes in 0 blocks
==4866==    still reachable: 72,704 bytes in 1 blocks
==4866==         suppressed: 0 bytes in 0 blocks
==4866== Rerun with --leak-check=full to see details of leaked memory
==4866==
==4866== For counts of detected and suppressed errors, rerun with: -v
==4866== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
Congratulation, all done!
yuq8@andromeda-30 00:42:49 ~/253P/lab/lab1
$
```