Pengfei Li (pl189)

Homework 3, ECE 590 & CS320 Software Reliability.

What fraction of all executable statements in your program is executed? (Statement coverage)
90.5%

What fraction of all branches in your program are executed? (Branch coverage)
100%

What fraction of all functions in your program are called? (Function coverage)
100%

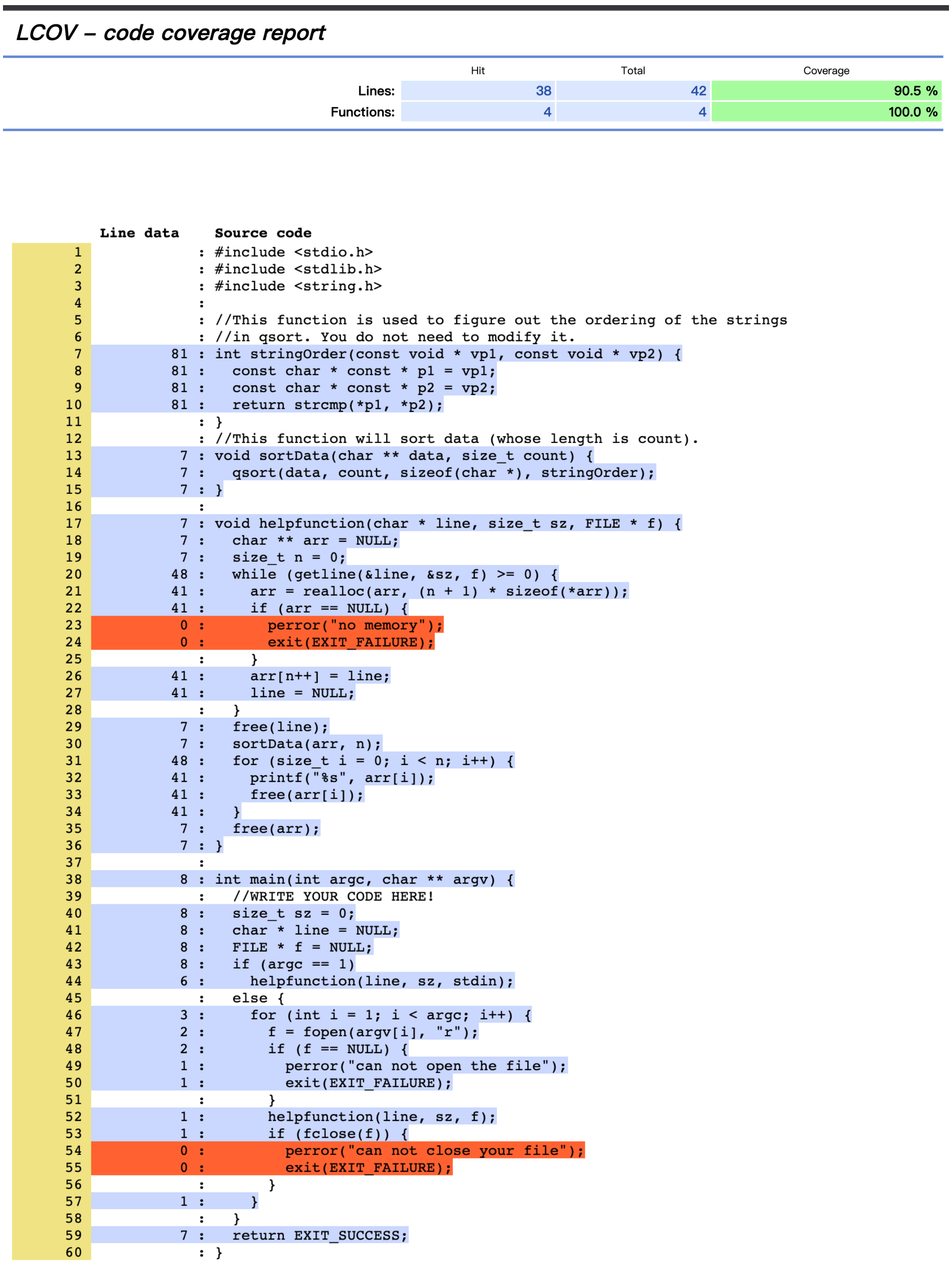
For 1, 2, and 3, if necessary, generate more test cases to have at least 90% coverage for each of the above metrics.

Identify parts of your code that are hard to cover and discuss why
Some corner case is hard to cover. For example, my program would exit when there is no memory, and the file cannot be closed. It is hard to go through these statements in my testcase because my test case can't be a file that can't be closed or my computer can't have no memory. Of course, it will happen in very rare cases.
In some complex projects, there may be more codes which are hard to cover, which may be caused by poor understanding of the design logic, high code repetition or blind spots in the test cases.

I also use 'lcov' to visualize the result of 'gcov' to make it clearer.

Result and detailed information about 'lcov' is on the next page

Result in lcov”

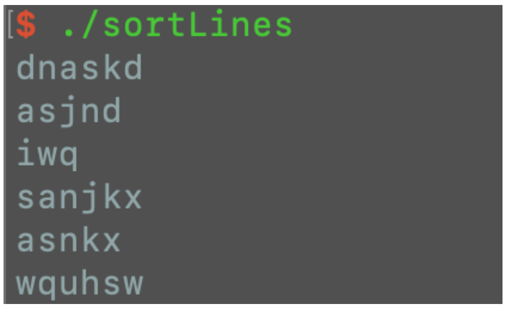


LCOV – code coverage report

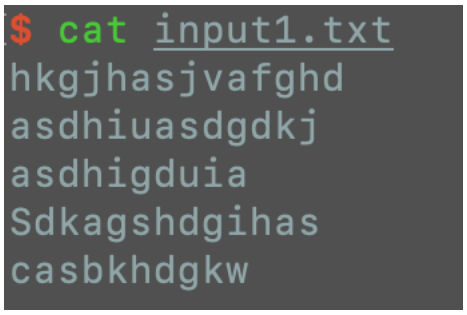
	Hit	Total	Coverage
Lines:	38	42	90.5 %
Functions:	4	4	100.0 %

```
 Line data    Source code
            : #include <stdio.h>
            : #include <stdlib.h>
            : #include <string.h>
            :
            : //This function is used to figure out the ordering of the strings
            : //in qsort. You do not need to modify it.
         81 : int stringOrder(const void * vp1, const void * vp2) {
         81 :   const char * const * p1 = vp1;
         81 :   const char * const * p2 = vp2;
         81 :   return strcmp(*p1, *p2);
            : }
            : //This function will sort data (whose length is count).
          7 : void sortData(char ** data, size_t count) {
          7 :   qsort(data, count, sizeof(char *), stringOrder);
          7 : }
            :
          7 : void helpfunction(char * line, size_t sz, FILE * f) {
          7 :   char ** arr = NULL;
          7 :   size_t n = 0;
         48 :   while (getline(&line, &sz, f) >= 0) {
         41 :     arr = realloc(arr, (n + 1) * sizeof(*arr));
         41 :     if (arr == NULL) {
          0 :       perror("no memory");
          0 :       exit(EXIT_FAILURE);
            :     }
         41 :     arr[n++] = line;
         41 :     line = NULL;
            :   }
          7 :   free(line);
          7 :   sortData(arr, n);
         48 :   for (size_t i = 0; i < n; i++) {
         41 :     printf("%s", arr[i]);
         41 :     free(arr[i]);
         41 :   }
          7 :   free(arr);
          7 : }
            :
          8 : int main(int argc, char ** argv) {
            :   //WRITE YOUR CODE HERE!
          8 :   size_t sz = 0;
          8 :   char * line = NULL;
          8 :   FILE * f = NULL;
          8 :   if (argc == 1)
          6 :     helpfunction(line, sz, stdin);
            :   else {
          3 :     for (int i = 1; i < argc; i++) {
          2 :       f = fopen(argv[i], "r");
          2 :       if (f == NULL) {
          1 :         perror("can not open the file");
          1 :         exit(EXIT_FAILURE);
            :       }
          1 :       helpfunction(line, sz, f);
          1 :       if (fclose(f)) {
          0 :         perror("can not close your file");
          0 :         exit(EXIT_FAILURE);
            :       }
          1 :     }
            :   }
          7 :   return EXIT_SUCCESS;
            : }
```

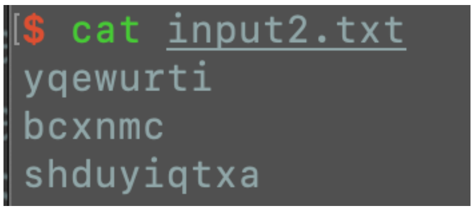
Test case:



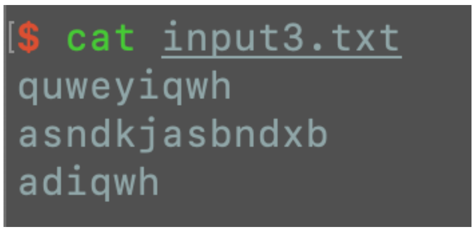
```
$ ./sortLines
dnaskd
asjnd
iwq
sanjkx
asnkx
wquhsw
```



```
$ cat input1.txt
hkgjhasjvafghd
asdhiuasdgdkj
asdhigduia
Sdkagshdgihas
casbkhdgkw
```

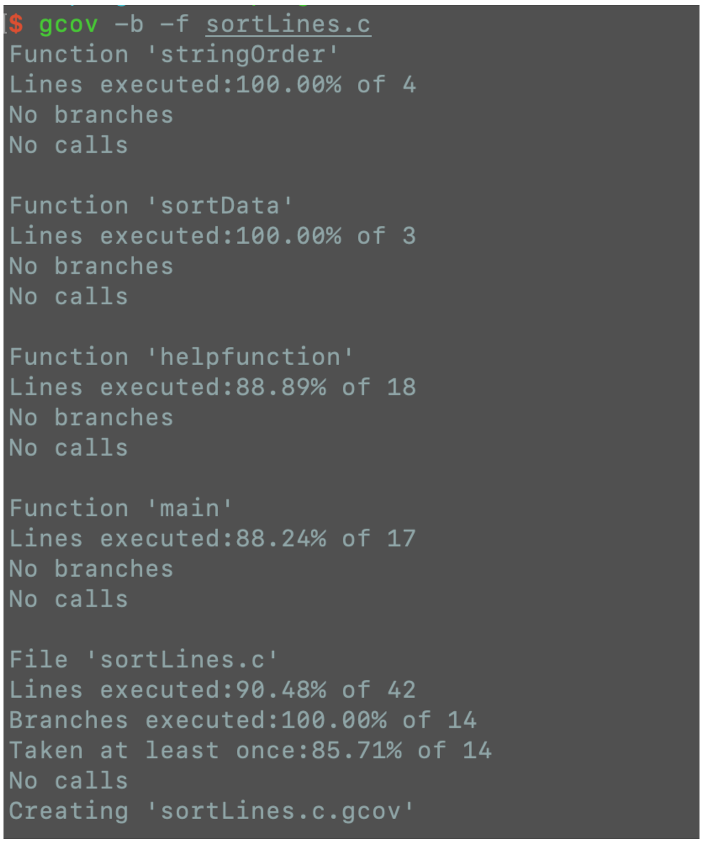


```
$ cat input2.txt
yqewurti
bcxnmc
shduyiqtxa
```



```
$ cat input3.txt
quweyiqwh
asndkjasbndxb
adiqwh
```

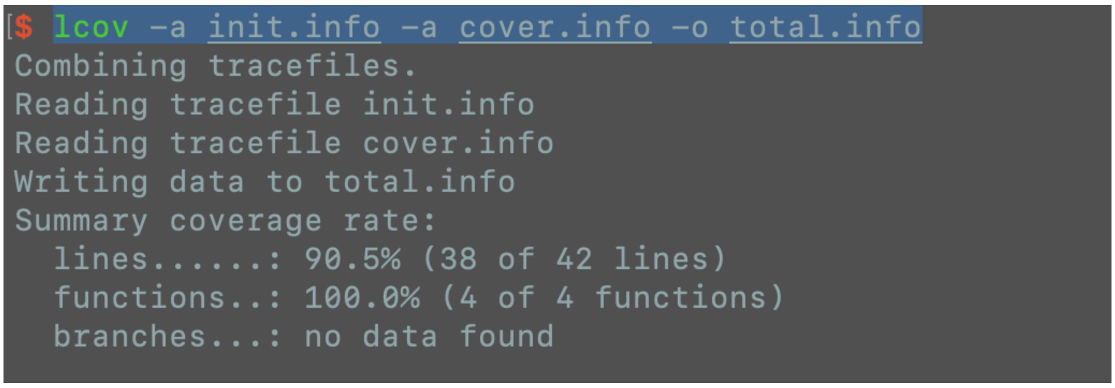
Result in gcov



```
$ gcov -b -f sortLines.c
Function 'stringOrder'
Lines executed:100.00% of 4
No branches
No calls

Function 'sortData'
Lines executed:100.00% of 3
No branches
No calls

Function 'helpfunction'
Lines executed:88.89% of 18
No branches
No calls

Function 'main'
Lines executed:88.24% of 17
No branches
No calls

File 'sortLines.c'
Lines executed:90.48% of 42
Branches executed:100.00% of 14
Taken at least once:85.71% of 14
No calls
Creating 'sortLines.c.gcov'
```

Lcov reference:
https://wiki.documentfoundation.org/Development/Lcov

Test procedures
in the directory of sorLines.c.gcov, we run:
lcov -c -d ./ -o cover.info
lcov -c -i -d ./ -o init.info
lcov -a init.info -a cover.info -o total.info
we get:



```
[$ lcov -a init.info -a cover.info -o total.info
Combining tracefiles.
Reading tracefile init.info
Reading tracefile cover.info
Writing data to total.info
Summary coverage rate:
  lines......: 90.5% (38 of 42 lines)
  functions..: 100.0% (4 of 4 functions)
  branches...: no data found
```

run : *genhtml -o ./ cover.info*
we get index.html and open it, it is about the coverage