Program #4 (TriePrediction): Final Report

Your final test case transcripts for Program #4 (TriePrediction) are now posted. You can access them by clicking over to **Program #4: TriePrediction (https://webcourses.ucf.edu/courses/1268969/assignments/5313578)** and, in the message from me that says "See attached file," clicking on "TriePrediction.c."

Test Cases

Attachment: ***program04-testcases.zip (https://webcourses.ucf.edu/courses/1268969/files/64246799/download?wrap=1)***

This archive contains the test cases used in grading this assignment. There are 21 test cases, each worth 5 points.

If you unzip the attached test case archive into a new directory and add your source file (TriePrediction.c) to that directory, you can use the test-all.sh script to compile and run the test cases like so:

```
bash test-all.sh
```

Alternatively, you can compile and run individual test cases manually. For standard test cases, which rely on command line input to specify a corpus and input file, you can compile and run like so:

```
gcc TriePrediction.c test_launcher_std.c
./a.out corpus_files/testcase01-corpus.txt test_cases/testcase01.txt > myoutput01.txt
diff myoutput01.txt sample_output/testcase01-output.txt
```

For unit tests, which override your main() function and make direct calls to individual required functions, you can compile and run at the command line like so:

```
cp test_cases/unit_test01.c .
gcc TriePrediction.c test_launcher_unit.c unit_test01.c
rm unit_test01.c
./a.out > myoutput01.txt
diff myoutput01.txt sample_output/unit_test01-output.txt
```

There are two valgrind tests. One runs your program in the standard fashion, with corpus and input file specified at the command line, and the other runs a unit test case. The valgrind tests can be compiled and run like so:

```
gcc TriePrediction.c test_launcher_std.c -g
valgrind --leak-check=yes ./a.out \
   corpus_files/valgrind_testcase01-corpus.txt \
   test_cases/valgrind_testcase01.txt \
```

```
   > myoutput01.txt
diff myoutput01.txt sample_output/valgrind_testcase01-output.txt
```

```
cp test_cases/valgrind_unit_test01.c .
gcc TriePrediction.c test_launcher_unit.c valgrind_unit_test01.c -g
valgrind --leak-check=yes ./a.out > myoutput01.txt
diff myoutput01.txt sample_output/valgrind_unit_test01-output.txt
```

Valgrind will of course tell you whether it found memory leaks or not. The phrase to look for is, "no leaks are possible."

Your output should match the output given in the solutions files exactly, with no extraneous output

Interpreting Your Test Case Transcript

See the following guide for help interpreting your test case transcript: **Interpreting Test Case Transcripts (https://webcourses.ucf.edu/courses/1268969/pages/interpreting-test-case-transcripts)**

General Submission Statistics:

Number of submissions:

225 (79.23% of students enrolled)

Number of programs that compiled:

204 (90.67% of submissions)

Number of programs that compiled and implemented difficultyRating() correctly:

202 (89.78% of submissions)

Number of programs that compiled and implemented hoursSpent() correctly:

200 (88.89% of submissions)

Number of submissions that were named TriePrediction.c:

222 (98.67% of submissions)

Grading Criteria

Max score possible: 105/100

```
50%   standard test cases (10 test cases, 5 pts each)
45%   unit tests (9 test cases, 5 pts each)
10%   valgrind test cases (2 test cases, 5 pts each)

Point deductions may be imposed for those who did not employ good coding style
(comments and whitespace) at the discretion of the grader.
```

Test Case Pass Rate

Following are the test case pass rates for this assignment.

Tests Passed	Number of Students
21/21	60 (26.67% of submissions)
20/21	10 (4.44% of submissions)
19/21	14 (6.22% of submissions)
18/21	8 (3.56% of submissions)
17/21	4 (1.78% of submissions)
16/21	10 (4.44% of submissions)
15/21	6 (2.67% of submissions)
14/21	5 (2.22% of submissions)
13/21	2 (0.89% of submissions)
12/21	3 (1.33% of submissions)
11/21	2 (0.89% of submissions)
10/21	4 (1.78% of submissions)
9/21	4 (1.78% of submissions)
8/21	0 (0.00% of submissions)
7/21	3 (1.33% of submissions)
6/21	5 (2.22% of submissions)
5/21	5 (2.22% of submissions)
4/21	9 (4.00% of submissions)
3/21	4 (1.78% of submissions)
2/21	44 (19.56% of submissions)
1/21	0 (0.00% of submissions)
0/21	23 (10.22% of submissions)

Average: 11.69 (out of 21) **Sample size:** 225

Valgrind Test Case Pass Rate

Following are the pass rates for each of the two valgrind tests. (See above for details on running these tests.)

Test Case	Number of Students
valgrind_testcase01.txt	93 (41.33% of submissions)
valgrind_unit_test01.c	82 (36.44% of submissions)

Difficulty Ratings

Following are the difficulty ratings you assigned to this problem.

Difficulty Rating	Number of students
[1.0, 2.0)	2 (0.99% of submissions)
[2.0, 3.0)	7 (3.47% of submissions)
[3.0, 4.0)	44 (21.78% of submissions)
[4.0, 5.0)	93 (46.04% of submissions)
[5.0, 5.0]	56 (27.72% of submissions)

Average: 4.15 (out of 5) **Sample size:** 202

Hours Spent

Following are the hours you invested in this program. I excluded two outliers from this summary (0.0 hours and 99.0 hours).

```
Hours Spent    # of students
(0.0, 1.0)     0   (0.00% of submissions)
[1.0, 2.0)     1   (0.50% of submissions)
[2.0, 3.0)     4   (2.00% of submissions)
[3.0, 4.0)     1   (0.50% of submissions)
[4.0, 5.0)     2   (1.00% of submissions)
[5.0, 7.5)     7   (3.50% of submissions)
[7.5, 10.0)    8   (4.00% of submissions)
[10.0, 15.0)   33  (16.50% of submissions)
[15.0, 20.0)   25  (12.50% of submissions)
[20.0, 30.0)   56  (28.00% of submissions)
[30.0, 40.0)   33  (16.50% of submissions)
[40.0, 50.0)   17  (8.50% of submissions)
[50.0, 60.0)   8   (4.00% of submissions)
[60.0, 70.0)   4   (2.00% of submissions)
[70.0, 85.0)   1   (0.50% of submissions)

 Average: 23.30 hours (Min: 1.0, Max: 84.0, Stddev: 13.76)  Sample size = 200
```

End of report.