Brendan Boone

WRITEUP:
Assignment 6: The Great Firewall of Santa Cruz: Bloom Filters, Linked Lists and Hash Tables

Discussion:

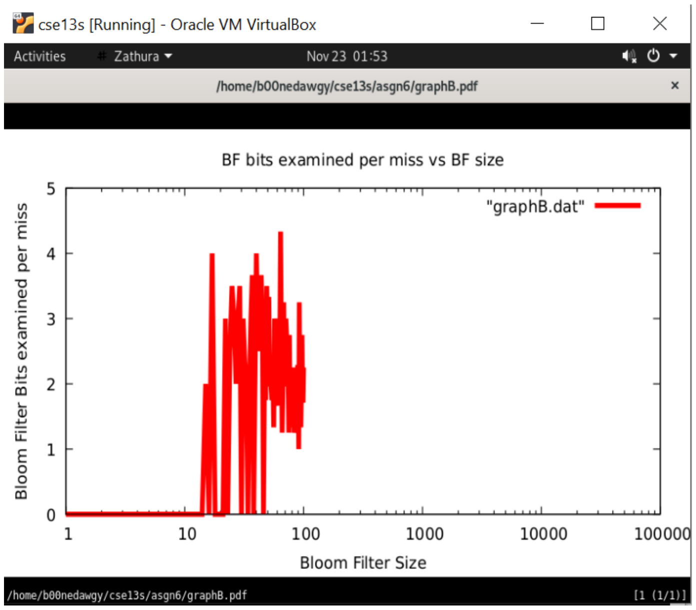
In this assignment, I learned a lot about how grading works in this class. Being able to tolerate management of the stats was very hard for me. The arithmetic was tedious and not fun for me. While checking my stats with the gitlab tests I found that the git lab tests are done using ‘diff’ between gitlab’s correct output and mine. I came across this discovery because I designed a bash script that used diff to compare my code with the given ./banhammer-dist file as well. I am upset that the example executable and the assignment pdf changed so much during this assignment’s two week period, but I’m glad I finally got it done. Working with the parsing function was a new experience for me. I didn’t expect to have to debug it as many times as I did. I learned a little bit about bits vectors and linked lists, and got to implement bit management for the first time as well. The most important thing I learned from this assignment was the ‘diff’ command.

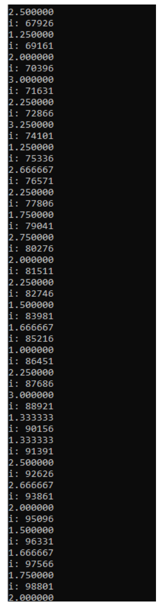
I made one main function: banhammer.c. The rest (6 if I recall) are files that hold a library of functions that I can use at my disposal inside of banhammer.c. Bf.c created a bloom filter that would store values so that whenever I accessed it, it would return a guaranteed boolean value whether the value I accessed was not inside it. Ht.c was used to implement a hash table so whenever the bloom filter thought the word I was looking for was inside it, I could confirm. Bv.c was to implement bit vectors to be used inside of the bloom filter. Ll.c was to implement Linked Lists to be used inside of the Hash table. Node.c was to implement nodes for the linked lists. Parser.c was to parse out every valid word found inputted into banhammer.c.

I had many obstacles in this assignment. The first was my parser file. It would not parse out words correctly as the gitlab tests were wanting. I did not understand that the parser would break after finding an invalid character. My initial parser file would skip any invalid character and return a word constructed from only valid characters before a space or newline. After looking through the class discord, eventually I discovered how the parser file should actually function. I fixed my parser to break on any occurrence of invalid characters. My next problem was the hardest to solve. My ht_probe could never align with the ‘-dist’ version. No matter how perfect my code could function, if my ht_probe did not return the correct output, it seemed all the gitlab test cases would fail. I would rewrite my code in many slight variations to see if it would correct itself, but it never did. I found the solution to my problem using three methods. While I had rewritten the code, I defined my newspeak.txt before the badspeak.txt in my bloom filter and hash table. However this was incorrect according to the assignment. One thing that helped me fix my problem was making sure the badspeak was defined before the newspeak. The newspeak would not overwrite the badspeak. I didn't understand why this was the case, according to the GPRSC context around this assignment, but it did fix my problem. The other method that helped me a lot was printing out the links traversed every time ht_lookup was called. It was much easier to see what was going wrong. My last method to find the solution to the ht_probe problem was making the diff script. I made several test files that were inputted into ./banhammer and -dist. The combination of the diff script and prints of links traversed per look up helped me over the edge to finish this assignment.

Questions and Answers:

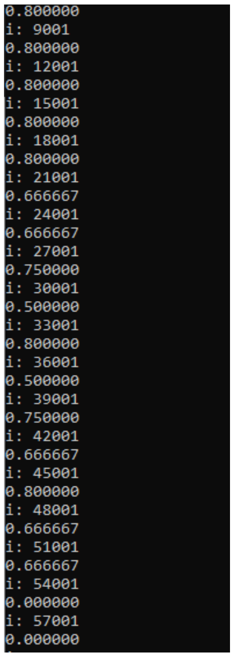
a) How does the number of Bloom filter bits examined per miss vary (for the same input) as the Bloom filter varies in size?
   i) As the Bloom Filter varies in size, the larger it gets the easier it is for bloom filter bits that were examined to miss. The graph and bash results below are used on the same echo stdin "hello young world! Isn't o-ld"



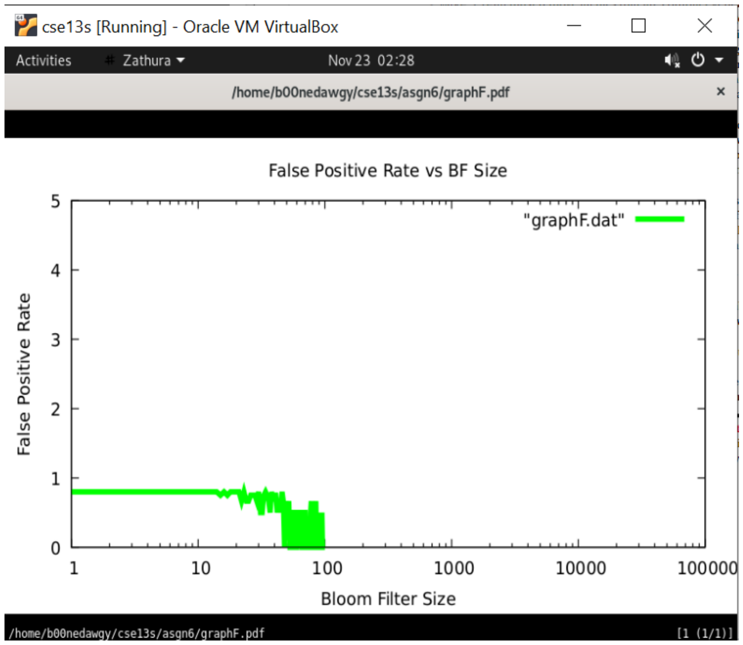


```
2.500000
i: 67926
1.250000
i: 69161
2.000000
i: 70396
3.000000
i: 71631
2.250000
i: 72866
3.250000
i: 74101
1.250000
i: 75336
2.666667
i: 76571
2.250000
i: 77806
1.750000
i: 79041
2.750000
i: 80276
2.000000
i: 81511
2.250000
i: 82746
1.500000
i: 83981
1.666667
i: 85216
1.000000
i: 86451
2.250000
i: 87686
3.000000
i: 88921
1.333333
i: 90156
1.333333
i: 91391
2.500000
i: 92626
2.666667
i: 93861
2.000000
i: 95096
1.500000
i: 96331
1.666667
i: 97566
1.750000
i: 98801
2.000000
```

b) How does changing the Bloom filter size affect the number of lookups performed in the hash table?
   i) As the Bloom filter grows in size the occurrence of false positives decreases in rate. The graph and bash show the input "hello young world! Isn't o-ld"'s false positives to decrease as the Bloom filter size increases. This means the larger the Bloom filter size, the more accurate it is because the hash table lookups return less misses.



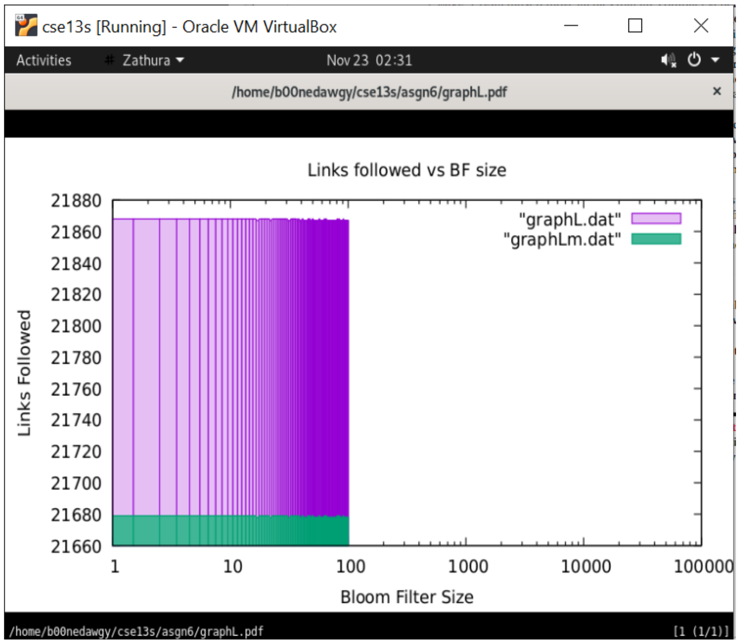
```
0.800000
i: 9001
0.800000
i: 12001
0.800000
i: 15001
0.800000
i: 18001
0.800000
i: 21001
0.666667
i: 24001
0.666667
i: 27001
0.750000
i: 30001
0.500000
i: 33001
0.800000
i: 36001
0.500000
i: 39001
0.750000
i: 42001
0.666667
i: 45001
0.800000
i: 48001
0.666667
i: 51001
0.666667
i: 54001
0.000000
i: 57001
0.000000
```

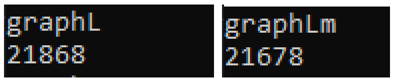


\

c) How does the number of links followed without the move-to-front rule compare to the number followed with the move-to-front rule?

i) The number of links followed without the move-to-front rule is larger than with the move-to-front rule. The move-to-front rule is implemented so searching through linked lists takes less traversal time, so it makes sense.



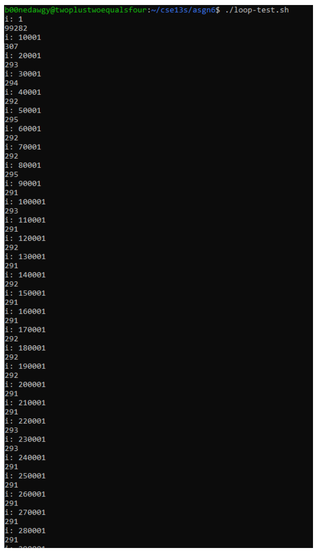


```
graphL
21868
```

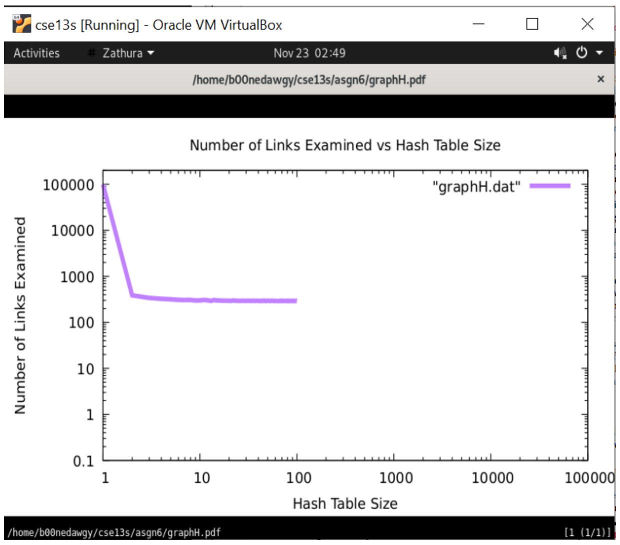
```
graphLm
21678
```

d) How does the number of links examined vary as the size of the hash table varies?

i) The number of links examined inversely approaches a certain value the larger the hash table gets. In my test, I used newspeak.txt. The larger my hash table got, the less difference there was from its previous number of links examined. Mine leveled at 291.

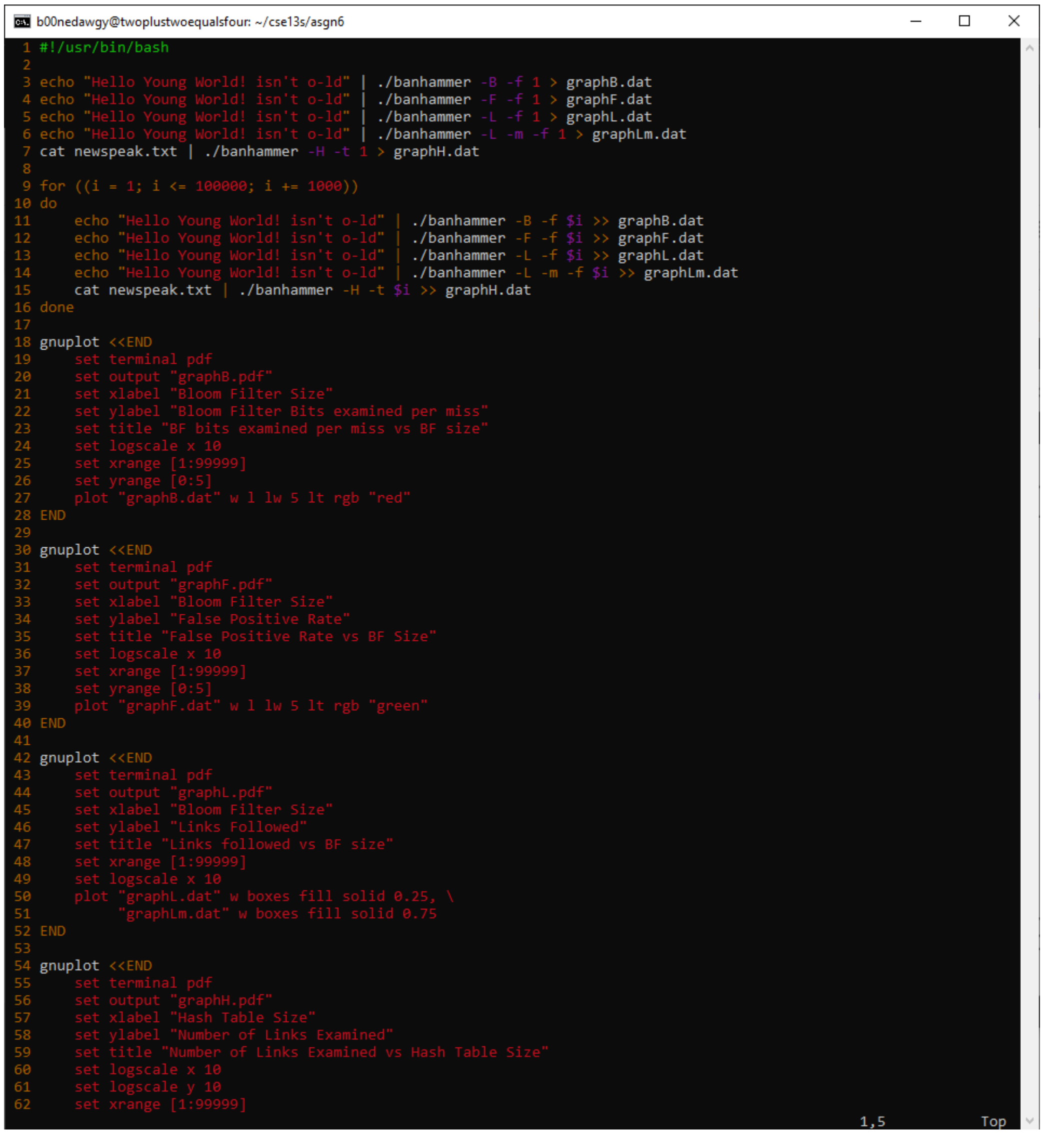


```
b00nedawgy@twoplustwoequalsfour:~/cse13s/asgn6$ ./loop-test.sh
i: 1
99282
i: 10001
307
i: 20001
293
i: 30001
294
i: 40001
292
i: 50001
295
i: 60001
292
i: 70001
292
i: 80001
295
i: 90001
291
i: 100001
293
i: 110001
291
i: 120001
292
i: 130001
291
i: 140001
292
i: 150001
291
i: 160001
291
i: 170001
292
i: 180001
292
i: 190001
292
i: 200001
291
i: 210001
291
i: 220001
293
i: 230001
293
i: 240001
291
i: 250001
291
i: 260001
291
i: 270001
291
i: 280001
291
```



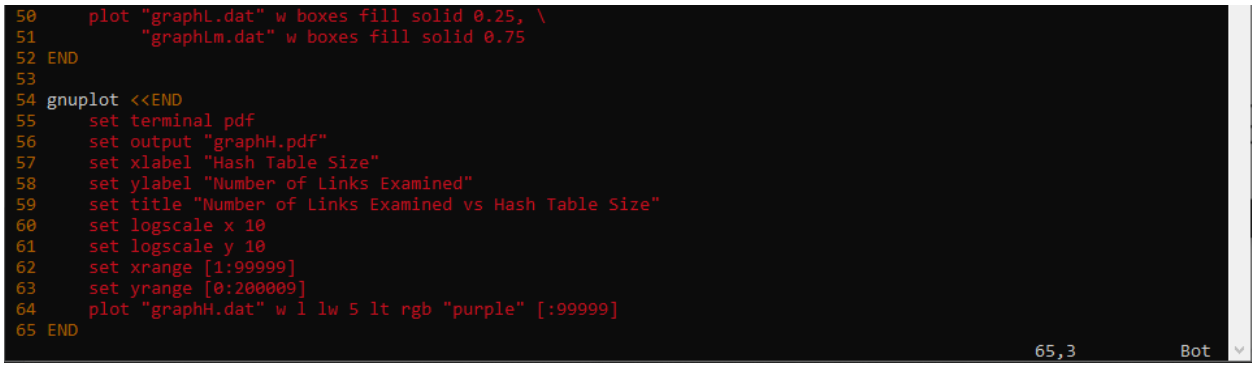
My Bash Script and runs:

Brendan Boone



b00nedawgy@twoplustwoequalsfour: ~/cse13s/asgn6

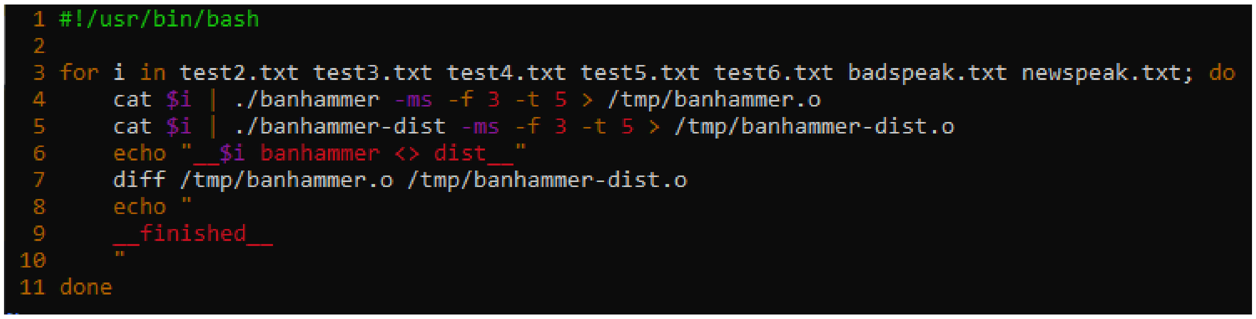
```
#!/usr/bin/bash

echo "Hello Young World! isn't o-ld" | ./banhammer -B -f 1 > graphB.dat
echo "Hello Young World! isn't o-ld" | ./banhammer -F -f 1 > graphF.dat
echo "Hello Young World! isn't o-ld" | ./banhammer -L -f 1 > graphL.dat
echo "Hello Young World! isn't o-ld" | ./banhammer -L -m -f 1 > graphLm.dat
cat newspeak.txt | ./banhammer -H -t 1 > graphH.dat

for ((i = 1; i <= 100000; i += 1000))
do
    echo "Hello Young World! isn't o-ld" | ./banhammer -B -f $i >> graphB.dat
    echo "Hello Young World! isn't o-ld" | ./banhammer -F -f $i >> graphF.dat
    echo "Hello Young World! isn't o-ld" | ./banhammer -L -f $i >> graphL.dat
    echo "Hello Young World! isn't o-ld" | ./banhammer -L -m -f $i >> graphLm.dat
    cat newspeak.txt | ./banhammer -H -t $i >> graphH.dat
done

gnuplot <<END
    set terminal pdf
    set output "graphB.pdf"
    set xlabel "Bloom Filter Size"
    set ylabel "Bloom Filter Bits examined per miss"
    set title "BF bits examined per miss vs BF size"
    set logscale x 10
    set xrange [1:99999]
    set yrange [0:5]
    plot "graphB.dat" w l lw 5 lt rgb "red"
END

gnuplot <<END
    set terminal pdf
    set output "graphF.pdf"
    set xlabel "Bloom Filter Size"
    set ylabel "False Positive Rate"
    set title "False Positive Rate vs BF Size"
    set logscale x 10
    set xrange [1:99999]
    set yrange [0:5]
    plot "graphF.dat" w l lw 5 lt rgb "green"
END

gnuplot <<END
    set terminal pdf
    set output "graphL.pdf"
    set xlabel "Bloom Filter Size"
    set ylabel "Links Followed"
    set title "Links followed vs BF size"
    set xrange [1:99999]
    set logscale x 10
    plot "graphL.dat" w boxes fill solid 0.25, \
         "graphLm.dat" w boxes fill solid 0.75
END

gnuplot <<END
    set terminal pdf
    set output "graphH.pdf"
    set xlabel "Hash Table Size"
    set ylabel "Number of Links Examined"
    set title "Number of Links Examined vs Hash Table Size"
    set logscale x 10
    set logscale y 10
    set xrange [1:99999]
                                                                1,5           Top
```



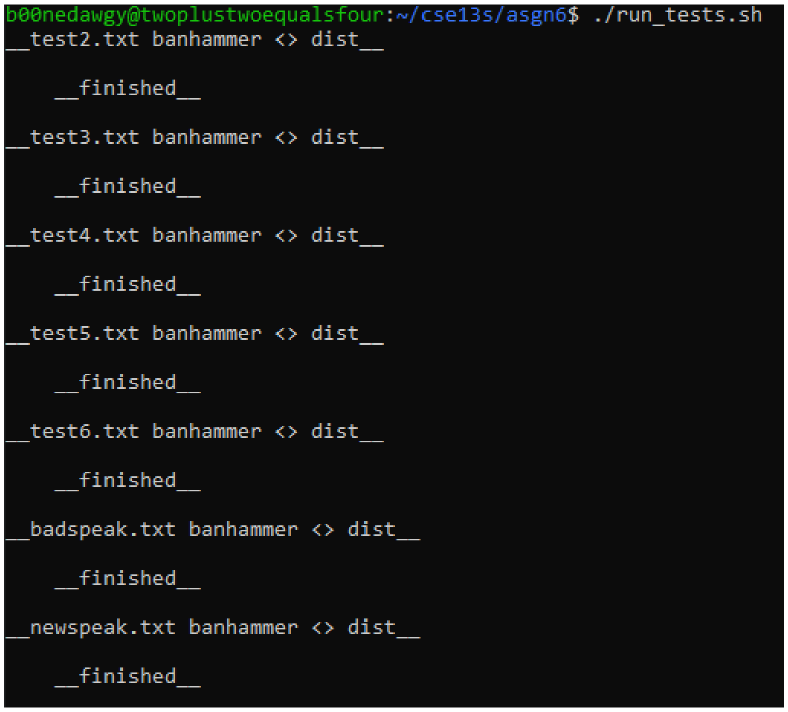
```
    plot "graphL.dat" w boxes fill solid 0.25, \
         "graphLm.dat" w boxes fill solid 0.75
END

gnuplot <<END
    set terminal pdf
    set output "graphH.pdf"
    set xlabel "Hash Table Size"
    set ylabel "Number of Links Examined"
    set title "Number of Links Examined vs Hash Table Size"
    set logscale x 10
    set logscale y 10
    set xrange [1:99999]
```

```
    set yrange [0:200009]
    plot "graphH.dat" w l lw 5 lt rgb "purple" [:99999]
END
                                                                65,3          Bot
```

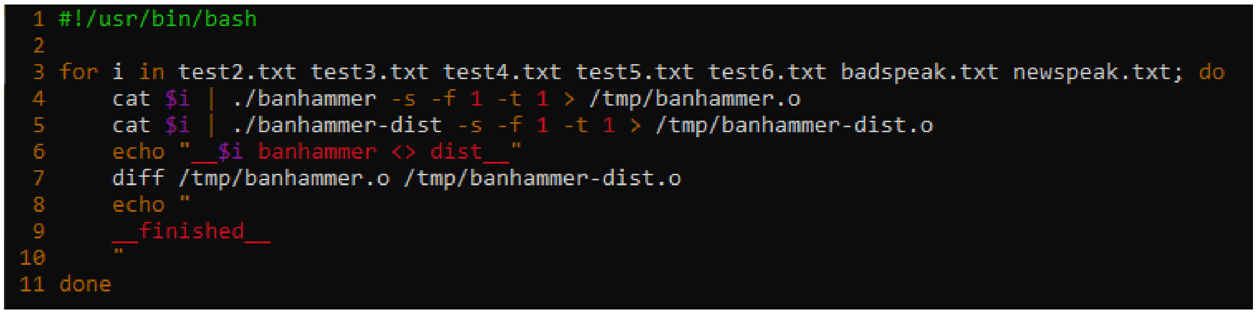
Brendan Boone



```
#!/usr/bin/bash

for i in test2.txt test3.txt test4.txt test5.txt test6.txt badspeak.txt newspeak.txt; do
    cat $i | ./banhammer -ms -f 3 -t 5 > /tmp/banhammer.o
    cat $i | ./banhammer-dist -ms -f 3 -t 5 > /tmp/banhammer-dist.o
    echo "__$i banhammer <> dist__"
    diff /tmp/banhammer.o /tmp/banhammer-dist.o
    echo "
    __finished__
    "
done
```

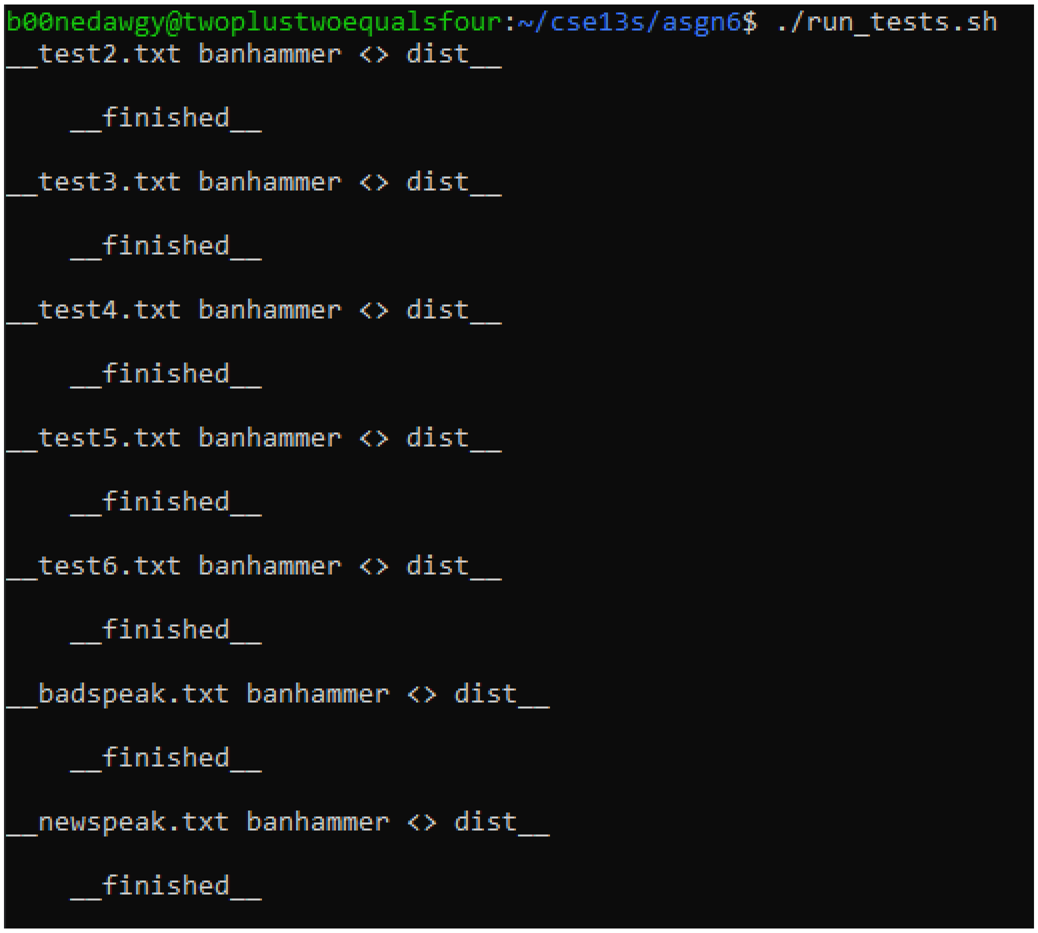


```
b00nedawgy@twoplustwoequalsfour:~/cse13s/asgn6$ ./run_tests.sh
__test2.txt banhammer <> dist__

    __finished__

__test3.txt banhammer <> dist__

    __finished__

__test4.txt banhammer <> dist__

    __finished__

__test5.txt banhammer <> dist__

    __finished__

__test6.txt banhammer <> dist__

    __finished__

__badspeak.txt banhammer <> dist__

    __finished__

__newspeak.txt banhammer <> dist__

    __finished__
```

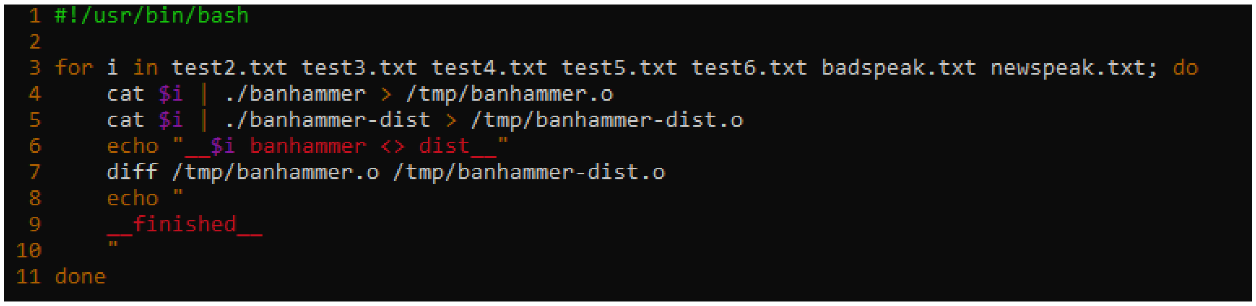


```
#!/usr/bin/bash

for i in test2.txt test3.txt test4.txt test5.txt test6.txt badspeak.txt newspeak.txt; do
    cat $i | ./banhammer -s -f 1 -t 1 > /tmp/banhammer.o
    cat $i | ./banhammer-dist -s -f 1 -t 1 > /tmp/banhammer-dist.o
    echo "__$i banhammer <> dist__"
    diff /tmp/banhammer.o /tmp/banhammer-dist.o
    echo "
    __finished__
    "
done
```

Brendan Boone

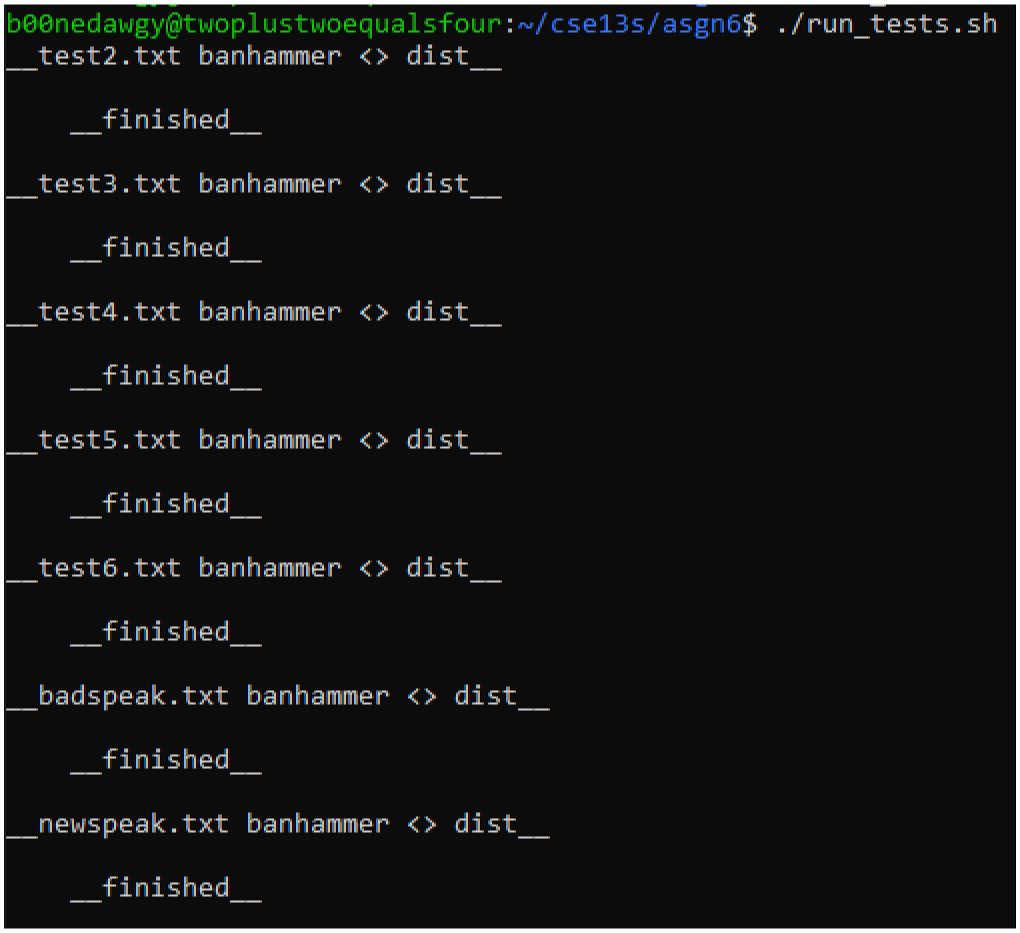


```
b00nedawgy@twoplustwoequalsfour:~/cse13s/asgn6$ ./run_tests.sh
__test2.txt banhammer <> dist__

    __finished__

__test3.txt banhammer <> dist__

    __finished__

__test4.txt banhammer <> dist__

    __finished__

__test5.txt banhammer <> dist__

    __finished__

__test6.txt banhammer <> dist__

    __finished__

__badspeak.txt banhammer <> dist__

    __finished__

__newspeak.txt banhammer <> dist__

    __finished__
```



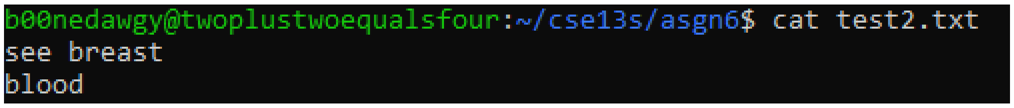
```
#!/usr/bin/bash

for i in test2.txt test3.txt test4.txt test5.txt test6.txt badspeak.txt newspeak.txt; do
    cat $i | ./banhammer > /tmp/banhammer.o
    cat $i | ./banhammer-dist > /tmp/banhammer-dist.o
    echo "__$i banhammer <> dist__"
    diff /tmp/banhammer.o /tmp/banhammer-dist.o
    echo "
    __finished__
    "
done
```

Brendan Boone

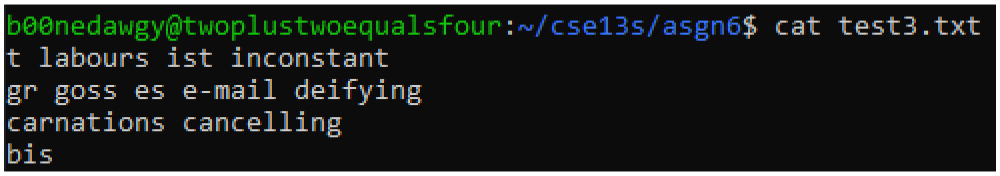


```
b00nedawgy@twoplustwoequalsfour:~/cse13s/asgn6$ ./run_tests.sh
__test2.txt banhammer <> dist__

    __finished__

__test3.txt banhammer <> dist__

    __finished__

__test4.txt banhammer <> dist__

    __finished__

__test5.txt banhammer <> dist__

    __finished__

__test6.txt banhammer <> dist__

    __finished__

__badspeak.txt banhammer <> dist__

    __finished__

__newspeak.txt banhammer <> dist__

    __finished__
```

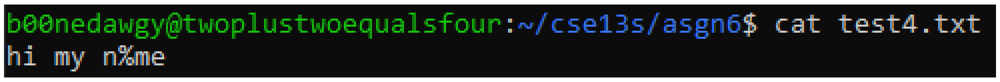
My Test Files:



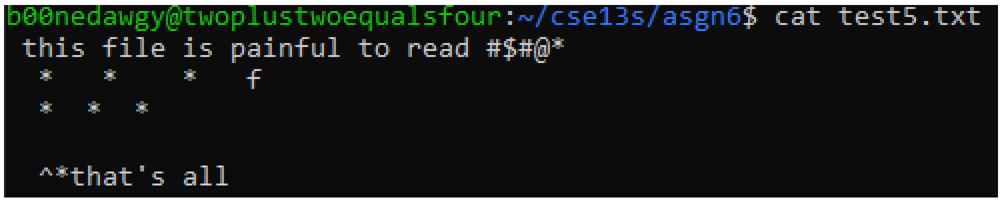
```
b00nedawgy@twoplustwoequalsfour:~/cse13s/asgn6$ cat test2.txt
see breast
blood
```



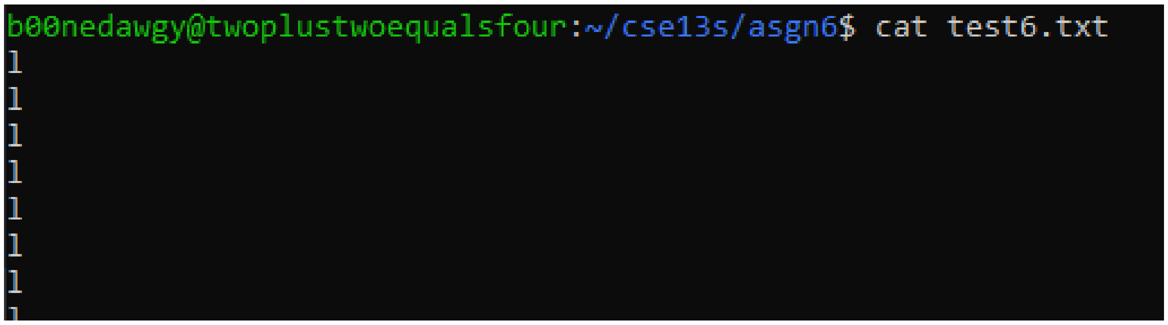
```
b00nedawgy@twoplustwoequalsfour:~/cse13s/asgn6$ cat test3.txt
t labours ist inconstant
gr goss es e-mail deifying
carnations cancelling
bis
```



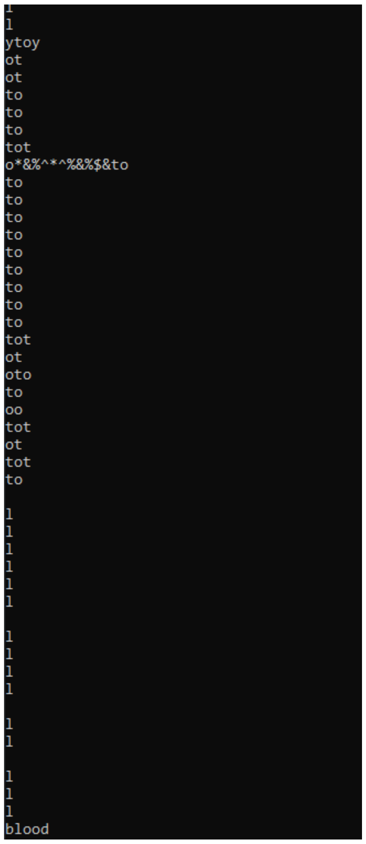
```
b00nedawgy@twoplustwoequalsfour:~/cse13s/asgn6$ cat test4.txt
hi my n%me
```



```
b00nedawgy@twoplustwoequalsfour:~/cse13s/asgn6$ cat test5.txt
 this file is painful to read #$#@*
  *   *    *   f
  *  *  *

  ^*that's all
```



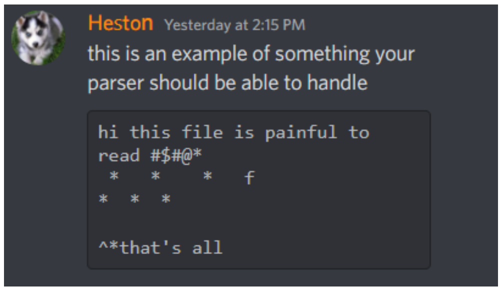
```
b00nedawgy@twoplustwoequalsfour:~/cse13s/asgn6$ cat test6.txt
l
l
l
l
l
l
l
```



```
l
ytoy
ot
ot
to
to
to
tot
o*&%^*^%&%$&to
to
to
to
to
to
to
to
to
to
tot
ot
oto
to
oo
tot
ot
tot
to

l
l
l
l
l
l

l
l
l
l

l
l

l
l
l
blood
```

Citations:

- https://stackoverflow.com/questions/47981/how-do-i-set-clear-and-toggle-a-single-bit
    - While making design - to set and clr bit
- https://stackoverflow.com/questions/23555267/member-reference-base-type-nodeint-is-not-a-structure-or-union
    - Help with node file -> delete function
- https://teaching.ethanmiller.org/cse13s/fall22/cse13s-fall22-lecture13.pdf
    - Help with getting filter index from hash
- https://stackoverflow.com/questions/17307275/what-is-the-printf-format-specifier-for-bool
    - Help printing true or false values
- https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html
    - Ascii table
- My test cases were inspired heavily by discussions in the class discord. The discord message I want to cite specifically was made by: Heston

- 

Heston Yesterday at 2:15 PM

this is an example of something your parser should be able to handle

```
hi this file is painful to
read #$#@*
 *   *    *   f
*  *  *

^*that's all
```

- I was able to figure out how to fix the functionality of my code at the end because of this post on piazza.
  - https://piazza.com/class/l8ahj4fji3i4om/post/512
  - Sameer Dash. Thanks.