asgn5

William Zhou

February 2024

1. **In part I, you implemented garbage collection routines that clean up dynamically-allocated memory. How did you make sure the memory was all cleaned up? How did you check?** I made sure the memory was all cleaned up by using the debugging tool valgrind. Valgrind is able to go through my code and give me a report of memory leaks and other memory issues.

2. **In Part II, you made a major optimization to the linked list optimization. What was it, and why do you think it changed the performance of bench1 so much?** I have not implemented it yet so I don't know.

3. **In Part III, you implemented hash tables. What happens to the performance of bench2 as you vary the number of buckets in your hash table? How many buckets did you ultimately choose to use?** I haven't implemented hash tables yet, but I know that if I increase the number of buckets in my hash table, it will take up more memory. However, having more buckets means that there will be less collisions, which will in turn speed up my program. I have not chosen how many buckets I will use yet as I do not have the implementation finished. As memory is not an issue for my computer, I will probably increase the bucket count until the performance increase is negligible.

4. **How did you make sure that your code was free from bugs? What did you test, and how did you test it? In particular, how did you create inputs and check the output of uniqq?**

   (a) I haven't finished yet, but I would make sure my code was free from bugs by running tests and confirming that my output matches my expectations. I can create sample inputs that test for various cases to ensure my code works the way I want it to.

   (b) For uniqq specifically, I can once again create sample inputs that test for various cases. I can check the output by returning the output into a textfile and reading the results. For example:

      i. Test how it would behave if all lines were the same with this input (should return 0): Fds

Fds
Fds
Fds
Fds

ii. Test how it would behave if all lines were unique with this input(should return 5): Hi
Yo
Joe
Bob
Sally

iii. Test how it would behave if lines have same string but varying capitalization(should return 5): Hi
hi
Joe
joE
jOe