Chase Badalato 101072570

Computation of Average Time:
The average times that I found are below, with an explanation under that.

Average of all execution times:
Average of an APPEND command:
Average of a DELETE command:
Average of a REMOVE command:
Average of a SEARCH command:

- I ran my program 10 times for each command, 40 times total.
- I would write down the time it took to do a command
- After I ran 1 command 10 times I would calculated the average of that specific commands time
- I would also run all 4 commands in 1 run, and repeat that 10 times.
- This would give me the average times.
- My average values make sense because the complexity of the code that is required to execute the commands is not complicated or long.

Compiling

```
tar -xzvf assignmentThree_101072570.tar.gz
make
```

To run:

```
./Text-Manager (must be ran first)
./User (must be ran second)
```

Files
Makefile
Text-Manager.c
User.c
linked_list.c
shared_items_def.h

Pseudo-code:
The pseudo code is as follows:

<u>User.c:</u>

```
Set up a connection with the input and output queue

While (true)
```

```
    {
    gather command choice from user
    gather text to use from user

    If (user wants to exit) {
        Delete the queues and exit
    }
    place command and data into a output struct
    send this data to the text manager

    receive response from text manager

    if (the ack value is a 1)
    {
        Command executed correctly
        Print the response that the text-manager sent if applicable to the
        command. (Ex. a search will return first sentence found, but an append
        doesn't need to return anything other then a ack value)
    }
    If (ack is a 0) {
        The command could not be executed.  The reasoning will be different for
        all 4 command types.
    }
}
```

<u>Text-Manager.c</u>

```
If (there is already queues existing) {
    We must delete them to ensure proper behavior
}

Set up a connection with the input and the output queues

While (true) {
    Receive a value from the User

    If (this gives an error and its error 43, no value found) {
        This means User closed queues, exit
    }
```

Call functions to manipulate a linked list based on the received command. For example an append command will call the append_list function that appends the inputted sentence into a linked list node. The linked list is the main way that I store my data

Based on the return value of the called functions create an outgoing packet to send back to the User in the form of an acknowledgement structure.

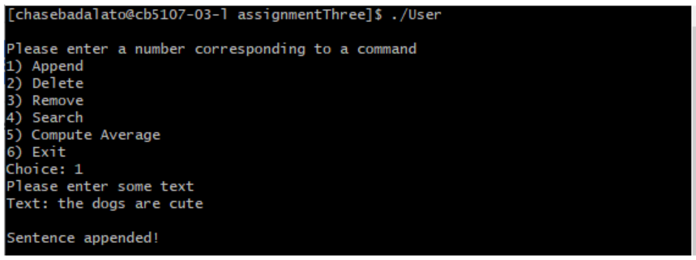
}

Testing:

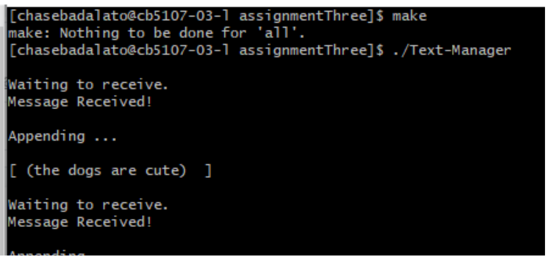
last time I was told to add screenshots to my testing, so I have added them below

I tested the 4 possible commands, testing with both successful and unsuccessful sentences and words. More explained after the images.

<u>Append Text</u>

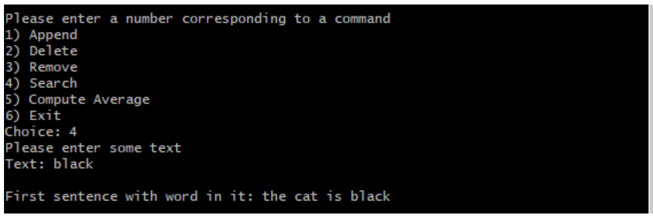


```
[chasebadalato@cb5107-03-l assignmentThree]$ ./User

Please enter a number corresponding to a command
1) Append
2) Delete
3) Remove
4) Search
5) Compute Average
6) Exit
Choice: 1
Please enter some text
Text: the dogs are cute

Sentence appended!
```

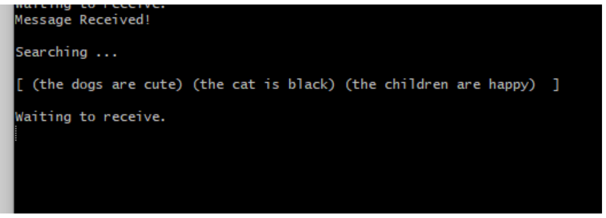


```
[chasebadalato@cb5107-03-l assignmentThree]$ make
make: Nothing to be done for 'all'.
[chasebadalato@cb5107-03-l assignmentThree]$ ./Text-Manager

Waiting to receive.
Message Received!

Appending ...

[ (the dogs are cute)  ]

Waiting to receive.
Message Received!
```

<u>Search for a Word within the sentences</u>

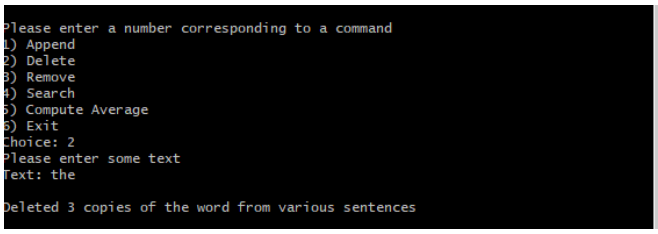


```
Please enter a number corresponding to a command
1) Append
2) Delete
3) Remove
4) Search
5) Compute Average
6) Exit
Choice: 4
Please enter some text
Text: black

First sentence with word in it: the cat is black
```

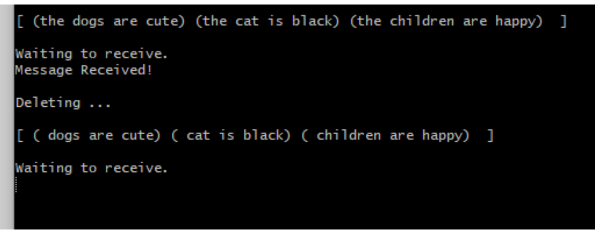


```
Message Received!

Searching ...

[ (the dogs are cute) (the cat is black) (the children are happy)  ]

Waiting to receive.
```

<u>Delete all instances of the given word</u>

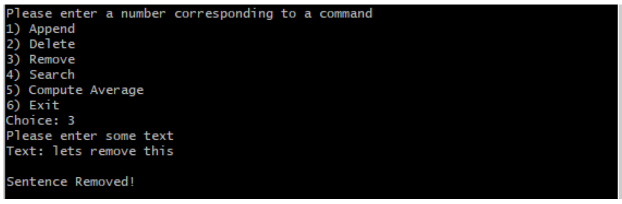


```
lease enter a number corresponding to a command
) Append
) Delete
) Remove
) Search
) Compute Average
) Exit
hoice: 2
lease enter some text
ext: the

eleted 3 copies of the word from various sentences
```

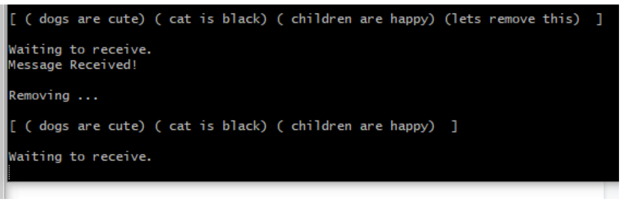


```
[ (the dogs are cute) (the cat is black) (the children are happy)  ]

Waiting to receive.
Message Received!

Deleting ...

[ ( dogs are cute) ( cat is black) ( children are happy)  ]

Waiting to receive.
```

<u>Remove an entire sentence</u>

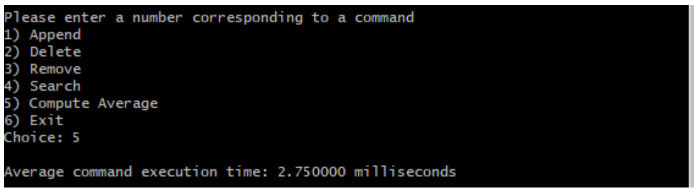


```
Please enter a number corresponding to a command
1) Append
2) Delete
3) Remove
4) Search
5) Compute Average
6) Exit
Choice: 3
Please enter some text
Text: lets remove this

Sentence Removed!
```

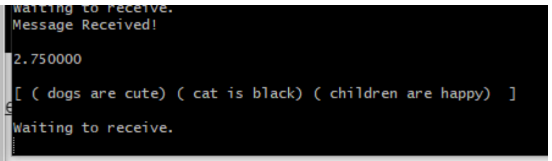


```
[ ( dogs are cute) ( cat is black) ( children are happy) (lets remove this)  ]

Waiting to receive.
Message Received!

Removing ...

[ ( dogs are cute) ( cat is black) ( children are happy)  ]

Waiting to receive.
```

<u>Calculate the average time</u>

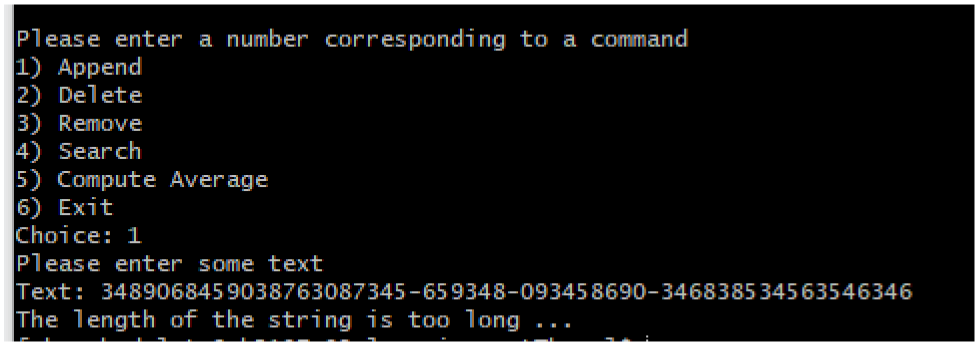


```
Please enter a number corresponding to a command
1) Append
2) Delete
3) Remove
4) Search
5) Compute Average
6) Exit
Choice: 5

Average command execution time: 2.750000 milliseconds
```



```
Waiting to receive.
Message Received!

2.750000

[ ( dogs are cute) ( cat is black) ( children are happy)  ]

Waiting to receive.
```

<u>Inputted text is more than 35 chars long</u>



```
Please enter a number corresponding to a command
1) Append
2) Delete
3) Remove
4) Search
5) Compute Average
6) Exit
Choice: 1
Please enter some text
Text: 348906845903876308734 5-659348-093458690-3468385345635463 46
The length of the string is too long ...
```

On top of the tests that I ran above, I also ran multiple tests with all of the different commands, in different orders. I had run tests that I know would not be successful such as trying to search, remove, or delete values when nothing has been appended yet. I also tried searching, removing, and deleting values when the list was populated but the values were not in the list. All of these results were properly handled with my error handling.

My ack response struct allows for only the user program to need to be looked at by a user. Any information that is important for the user to see gets passed back from the text-manager to the user in a ack_struct.response variable.