Hash Count

This program processes a text file by reading its content, separating words using various delimiters and storing them into a hash table. The hash table uses dynamically allocated linked lists, where the words are hashed based on the sum of their character values modulo a chosen hash value. Additional functionalities include word occurrence counting, alphabetic ordering, selective bucket output, filtering based on buckets and storing the hash table in a binary file.

Compilation

Only the standart librarys were used except of the function getline(), altough it is prototyped in the stdio.h header, it is not part of c99. Therefore an additional #define is used at every beginning of a source file where the function appears.

```
#define _POSIX_C_SOURCE 200809L
```

A Makefile was also included for a friendllier usage. to compile and run the program with a pre selected .txt file just type in the terminal:

```
make run
```

If you want to use your own file just type:

```
make
```

Then you can use the executable like this:

```
./execute -f YOUR_FILE_NAME_HERE.txt
```

If you want to clean up all the .o files use the clean command if you also wish to delete the executable try fclean:

```
make clean
make fclean
```

Execution

This program is pretty straightforward. If you run it, the first output will be your hashtable with all the buckets and the hashed words in it. Then you can just follow the instructions it gives you. NOW GO AND HAVE FUN!... But to be honest it is not that much fun i guess... atleast it is honest work!

Tasks

The tasks which were succesfully implemented are the following:

```
1. Main Task (50 points)

-> Open a text file and read the text row by row.
   Separate words by these characters:
```

-Space ( )

-Dot (.)

-new line (\n)

-> Store the values in a hashtable using dynamic linked lists.

-> Use the sum of character values as value to be hashed. E.g. Cat = (67+97+116) % 23

-> Words shall be stored in dynamically allocated memory

-> Select a good spreading hash value

-> Print out the hash in a readable way. E.g. max 10 words per row.

2. Task (15 points)

-> The occurrence of words shall be counted instead of multiple entries in the list. (see example output)

-> Add additional word separators:

-semicolon (;)

-colon (:)

-comma (,)

-question mark (?)

-tab (\t)

3. Task (15 points)

-> The lists shall always retain alphabetic order.

4. Task (15 points)

-> Give the user a choice to select one bucket to output separately. Print out the entries in the Bucket.

-> Let the user select one or more buckets and remove all words from the text which do not match to these selected buckets and write it to a new text file.

```
6. Task (15 points)


-> Store the hash in a binary file


7. Task (15 points)

-> The program arguments are handled by getopt(3).
```