1 Introduction

This assignment is designed:

- to refamiliarize you with C, the C compiler, and the UNIX system;
- to give you some practice with multiple file compilation; and
- to introduce you to Makefiles and the `make` command.

You are to write a C program that searches a dictionary for words that contain a given number of occurrences of a given string. Your program first reads in a list of words (*i.e.*, a dictionary) from a file. It then repeatedly reads in a string (the "pattern") and an integer (the "count") from the user. The count is the exact number of times the pattern must appear in a single dictionary word for a "match" to have occurred. The program finds all the words in the array that match in this way. It prints the first ten of these words; if more than ten words match, the number of additional matching words is printed. For example, if seventeen words match the pattern, ten words are printed followed by a message indicating that seven additional matches were not printed. Note that the number of occurrences includes overlaps. For example, there are five occurrences of the pattern `aa` in the string `aaaaaa`.

Unix systems come with a standard word list. On the UCS `gl` machines (`umbc8`, `umbc9`, `umbc10`, and the SGI workstations), this list is in the file `/usr/lib/dict/words`. The words are listed one–per–line with no extraneous white space. Some of the words are capitalized. Some contain the apostrophe character. The maximum word length is 22 letters. You should use `/usr/lib/dict/words` as your word list (although you may want to experiment with a much shorter list while you are developing your code). You can use the Unix command `wc` to find the number of words in the word list (see the `man` page for details).

Your program should read the dictionary file just once, loading its contents into an array (it must not re–read the dictionary for each word). A sample run of the program is given in Section 8. Your output does not have to be formatted exactly the same as that shown in the sample, but should be in a similar style.

2 Error Checking on Input

The program should terminate after issuing an error message if the dictionary file cannot be opened. Error messages (but not program termination) are to be issued for other erroneous input when:

- the count is less than or equal to zero;
- the count is missing or is not a number; or
- a non–alphabetic character is in the pattern.

Note that the program is to terminate when the user types `^D` (*i.e.*, on end–of–file); a blank input line is not to be considered an error.

3 File Structure

You must implement this homework using multiple source files. The main function should appear (with little else) in its own file (suggestion: name it `hw1.c`). Other functions must be declared in an interface file (suggestion: name it `hw1_aux.h`) and be defined in an implementation file (suggestion: name it `hw1_aux.c`). The interface file must be a **guarded** header file. Don't forget to place a comment at the top of the file containing your main program describing the help you received on the homework (as described in the Prospectus). If you received no help except from the course staff, the Help Center, or the ACS consultants, then the comment should indicate that no help was received. **Homework that does not contain this comment will be returned ungraded.**

4 Producing a Transcript

You will be submitting a transcript of your program in action. To make a transcript, enter the `script` command. A file named `typescript` will be created containing a transcript of your session. Run your program (by typing `hw1`). Be sure to enter at least as much data as is given in the sample transcript below, but don't overdo it. When you are finished, exit your program (by typing `^D`), then exit script by typing `exit`. The file `typescript` is now ready to submit.

5 Makefile

You must create and submit a Makefile that correctly compiles your homework upon issue of the `make` command with no arguments. You may name your files as you wish, but the executable created by the `make` command must be named `hw1`. Your Makefile will be used to compile your submitted code, so be sure it works. A simple Makefile that just does the compilation is fine.

6 Submission

You must submit the following files:

1. the "main function" file (`hw1.c`);
2. the interface file (`hw1_aux.h`);
3. the implementation file (`hw1_aux.c`);
4. your `Makefile`; and

5. a script (`typescript`) of your program execution.

To submit the files, use the `submit` program. For example, to submit the above files, enter:
`submit cs2021 hw1 hw1.c hw1_aux.h hw1_aux.c Makefile typescript`

You can check your submission by entering `submitls cs2021 hw1`.

7 Due Date

The homework is due on Thursday, 8 February. Submissions received by midnight of the due date will receive the on–time bonus. There is an automatic extension of two days, so submissions received by midnight of Saturday, 10 February will receive full credit. No submission will be accepted after Saturday, 10 February.

Do not wait till the last minute to submit your assignment. The automatic extension is given to accommodate unforeseen problems such as a machine or network crash. If you have not finished your assignment by the due date, please submit your work for partial credit. You can then use the automatic extension to submit improvements.

8 A Sample Run

Here is a transcript of a sample run.

```
Script started on Mon Jan 22 23:29:10 1996
umbc9[101] hw1
Enter string and occurrence_number: abc 1
abc
Babcock
Enter string and occurrence_number: a 5
No words found
Enter string and occurrence_number: a 4
amalgamate
extravaganza
Madagascar
Mahayana
paraphernalia
parliamentarian
Santayana
sarsaparilla
tarantara
```

```
Enter string and occurrence_number: a 3
academia
academician
acclamation
acquaintance
actuarial
adamant
adaptation
adiabatic
advantage
advantageous
There were 227 more words found
Enter string and occurrence_number: col 1
accolade
acolyte
Agricola
broccoli
bucolic
chocolate
col
cola
colander
colatitude
There were 73 more words found
Enter string and occurrence_number: Aa 1
Aaron
Enter string and occurrence_number: reco 1
irreconcilable
irrecoverable
precocious
precocity
recombinant
recompense
reconcile
recondite
reconnaissance
recovery
Enter string and occurrence_number: reco 2
No words found
Enter string and occurrence_number: co 0
---> Zero is not a valid occurrence number
Enter string and occurrence_number: $abc 1
---> Non-Alpha in string $abc
Enter string and occurrence_number: abc
---> So where's the number?
Enter string and occurrence_number: abc d
---> So where's the number?
Enter string and occurrence_number:
Enter string and occurrence_number:
Enter string and occurrence_number:
Enter string and occurrence_number: ^D
That's all folks
umbc9[102] exit
script done on Mon Jan 22 23:31:03 1996
```