Computation I 5EIA0
Homework 6: Concordance (v1.5, October 20, 2022)
Deadline Tuesday 25 October 2021 13:30

A concordance is a sorted list of words that occur in a text, together with the indices of where the words occur in the text. For example, the concordance for the text:

```
Dead-Sea scrolls were reverse engineered like this!
```

is

```
Concordance
  Dead-Sea: 0
engineered: 4
      like: 5
   reverse: 3
   scrolls: 1
     this!: 6
      were: 2
```

In this homework you will write a program to create concordances.

function	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	% per fn	cumulative %
quit	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	5%	5%
add word		1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	10%	15%
print concordance			1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	10%	25%
add index						1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	10%	35%
read file								1	1	1										1	15%	50%
remove word											1	1	1	1						1	20%	70%
find word at index															1	1					10%	80%
original text																	1				5%	85%
sort concordance																		1	1	1	15%	100%

Figure 1: Test cases and points per task when correctly implemented. Note that you do not have to implement all functions for subsequent tasks.

Important

- In this homework a predefined function is available for each task, *just like in the exams. It is a good idea to practice using the predefined functions to see how you should use them,* even if you don't use them for your final submission. For example `predefined_addIndex`, `predefined_printConcordance`. Therefore, if you get stuck on a task or want to skip it then you can use the predefined function instead of your own function.
- If you use the predefined function for a task anywhere in your code then you will not get points for all of the test cases of that task. For example, if instead of writing your own `addWord` in Task 1 you use `predefined_addWord` in later tasks then you will not get the points for test cases 2 and 3 that are unique for Task 1. You will get points for the other test cases, e.g. those that use `predefined_addWord`.
- To use the predefined functions you need to include `#include "predefined.h"` in your program. This is done automatically when you create a new .c file on Oncourse.
- The predefined functions only work on correct data structures. If an invalid concordance that does not follow the instructions is given to predefined functions then it may not work correctly. An attempt has been made to give error messages when this happens, but they cannot be exhaustive.

Task 1. Write a C program that asks the user to select the command that needs to be performed. The commands that need to be supported are listed in the following table:

command	operation
q	quit program
w	add word
p	print concordance
i	add index
r	read file
W	remove word
f	find word at index
o	print original text
s	sort concordance

In this task you only need to implement the quit command. In later tasks you will implement the remainder. Print the error message shown below if an invalid command is given:

```
Command? A
Unknown command 'A'
Command? q
Bye!
```

Your program must produce the *exact* output, including all spaces, punctuation, capitalisation, quotes, etc.

Hint: You can use the `" %c"` format string for `scanf`.

Task 2. The `predefined.h` file that you should include in your program with `#include "predefined.h"` contains the following declarations:

```
#define MAXWORDS 60
#define MAXINDEX 10
typedef struct _entry_t {
  char *word;
  int indices[MAXINDEX];
} entry_t;
extern void predefined_addWord(entry_t concordance[], char *word);
extern void predefined_printConcordance(entry_t concordance[]);
extern void predefined_addIndex(entry_t concordance[], char *word, int index);
```

No points will be deducted for including the predefined.h file, only for using the predefined functions.

Do not define the data type `struct _entry_t` since it is already defined in the predefined.h file. You will get a compilation error if you define it yourself too, and the predefined functions will not work.

Declare an array with MAXWORDS elements of type `entry_t` called `concordance` in your main function and initialise the word field of all elements to NULL. Write a function `void addWord(entry_t concordance[], char *word)` that adds the string `word` to the concordance. The concordance is an array of words ordered alphabetically. Initialise all entries in the `indices` array with -1. You must use `malloc` to use the minimum amount of space to store the new string. Do not use `strdup`. If the word is already in the concordance then do not insert or change the concordance. Do not give an error message either. If the concordance is full then give the error message `"Concordance is full"` and do not insert or change the concordance.

Add the 'w' command to the main function such that the user can add a word to the concordance. A word is any sequence of non-white-space characters of at most 10 characters and can be read using the `"%s"` format string for `scanf`. An example output is:

```
Command? w
Word? Pear
Command? w
Word? Apple
Command? w
Word? Apple
Command? q
Bye!
```

Hint: You'll probably get an error about a memory leak. Add a loop to free all malloc'd space in the quit command.

Your program should now pass test cases 1-2, and 3 after implementing the print concordance function. If you do not wish to implement the function `addWord` then you can use the `predefined_addWord` function, but you will not get the points for test cases 2-4. If you want to use `predefined_addWord` you can call it without adding any code in your program, since it has been declared in the predefined.h file.

Task 3. Write a function `void printConcordance(entry_t concordance[])` that prints all words in the concordance. Add the 'p' command in the main loop. When the concordance contains no words, the function should print a message that the concordance is empty as shown below. Note that the words are right aligned in 10 characters when printed. (Hint: see the field width in K&R Appendix B1.2.) You should also print the indices of the words. See the next task for an example output of that.

```
Command? p
The concordance is empty
Command? w
Word? Coconut
Command? p
Concordance
   Coconut:
Command? w
Word? Kiwi
Command? p
Concordance
   Coconut:
      Kiwi:
Command? w
Word? Banana
Command? p
Concordance
    Banana:
   Coconut:
      Kiwi:
Command? q
Bye!
```

Your program should now pass test cases 1-5. If you do not wish to implement the function `printConcordance` then you can use the `predefined_printConcordance` function, but you will not get the points for test cases 4-5.

Task 4. Write a function `void addIndex(entry_t concordance[], char *word, int index)` to add an index to a word in the concordance. Add the 'i' command in the main loop. Use the `indices` array in the `entry_t` structure to store the new index. Add the new index in the first empty position (with value -1) in the array. You may assume that the given index is at least zero and is inserted only once in the concordance; there is no need to check for this in your code. If the word is not in the concordance then give an error as shown below. If all entries in the `indices` array are already taken then exit the function without changing the concordance or printing a message. An example output is:

```
Command? w
Word? Apple
Command? i
Word index? Apple 2
Command? p
Concordance
     Apple: 2
Command? w
Word? Pear
Command? i
Word index? Pear 1
Command? i
Word index? Apple 3
Command? p
Concordance
     Apple: 2 3
      Pear: 1
Command? i
Word index? Lychee 100
Word Lychee not found
Command? q
Bye!
```

Your program should now pass test cases 1-7. If you do not wish to implement the function `addIndex` then you can use the `predefined_addIndex` function, but you will not get the points for test cases 6-7.

Task 5. Write a function `void readFile(entry_t concordance[], char *filename, int *index)` that inserts all words in the file with name `filename` in the concordance, starting at index `index`. Add the command 'r' to the main function that asks for a file name, calls readFile, and then prints the number of words that were inserted. The first word in the text file must be inserted in the concordance at index `index`, the next word at `index+1`, and so on.
Note that `index` is passed by reference. As a result, if you read multiple files then `index` keeps increasing (i.e. it is as if you read one long file). You can see this below when reading the test2 file twice. The first occurrence of Dead-Sea is at index 0 and 7 words are inserted in the concordance (indices 0-6). When reading the file again the Dead-Sea of the second file is inserted at index 7. Give an error message if the file cannot be opened.
When running your program you can test with the `test1`, `test2`, and `test3` files. These files are automatically placed in the same directory as your program when it is running on Oncourse. See the next page for the content of the three files.
This is an example of correct output:

```
Command? r
File name? test2
Inserted 7 words
Command? p
Concordance
  Dead-Sea: 0
engineered: 4
      like: 5
   reverse: 3
   scrolls: 1
     this!: 6
      were: 2
Command? r
File name? test2
Inserted 7 words
Command? p
Concordance
  Dead-Sea: 0 7
engineered: 4 11
      like: 5 12
   reverse: 3 10
   scrolls: 1 8
     this!: 6 13
      were: 2 9
Command? r
File name? no-file
Cannot open file no-file
Command? q
Bye!
```

Hint: Define `fileIndex` in the main function and pass it by reference to the function. It then records the largest index used by readFile.

Your program should now pass test cases 1-10. (There is no predefined function for this task.)

This is `test1`:

```
hello there
how are you
how is this file
is it there
it is here
```

This is `test2`:

```
Dead-Sea scrolls were reverse engineered like this!
```

This is `test3`:

```
function
quit
add word
print concordance
add index
read file
remove word
find word at index
original text
sort concordance
```

Task 6. Write a function `void removeWord(entry_t concordance[], char *word)` that removes the word from the concordance. Use the command 'W' to remove a word. Make sure to free all space that was malloc'd.

```
Command? w
Word? Union
Command? w
Word? Difference
Command? w
Word? Singleton
Command? i
Word index? Difference 1
Command? i
Word index? Difference 6
Command? i
Word index? Singleton 2
Command? i
Word index? Union 9
Command? i
Word index? Union 0
Command? p
Concordance
Difference: 1 6
 Singleton: 2
     Union: 9 0
Command? W
Word? Singleton
Command? p
Concordance
Difference: 1 6
     Union: 9 0
Command? W
Word? Difference
Command? p
Concordance
     Union: 9 0
Command? W
Word? Union
Command? p
The concordance is empty
Command? q
Bye!
```

Your program should now pass test cases 1-14. (There is no predefined function for this task.)

Task 7. Write a function `char *findWordAtIndex(entry_t concordance[], int index)` that returns a pointer to the word stored at index in the concordance. Return NULL if there is no word at that index. You can assume that there are no duplicate indices in the concordance. But there may be no word at that index, in which case the message `There is no word at index XX` (with XX replaced by the index) must be given. Add the 'f' command in the main loop with the behaviour shown below.

```
Command? w
Word? Apple
Command? i
Word index? Apple 2000
Command? i
Word index? Apple 2003
Command? i
Word index? Apple 1984
Command? w
Word? Samsung
Command? i
Word index? Samsung 2015
Command? w
Word? Oppo
Command? i
Word index? Oppo 2020
Command? p
Concordance
     Apple: 2000 2003 1984
      Oppo: 2020
   Samsung: 2015
Command? f
Index? 2003
The word at index 2003 is Apple
Command? f
Index? 1990
There is no word at index 1990
Command? q
Bye!
```

Your program should now pass test cases 1-16. (There is no predefined function for this task.)

Task 8. Write a function `void printOriginalText(entry_t concordance[])` that prints the original text from the concordance. Print words separated by a space all on one line. If there is no word at an index then print a question mark. You can see this in the output below, where there is no word at index 0 and index 2. You can assume that there are no duplicate indices in the concordance. Add the 'o' command in the main loop.

```
Command? o
Command? w
Word? first
Command? w
Word? third
Command? w
Word? fourth
Command? i
Word index? first 1
Command? i
Word index? third 3
Command? i
Word index? fourth 4
Command? p
Concordance
     first: 1
    fourth: 4
     third: 3
Command? o
? first ? third fourth
Command? q
Bye!
```

Hint: First find the maximum index in the concordance.

Your program should now pass test cases 1-17. (There is no predefined function for this task.)

Task 9. Write a function `void sortConcordance(entry_t concordance[])` that sorts the concordance on the first occurence of each word. Words with the same or no index (such as c and d below) are sorted alphabetically. You may assume the indices of each word are in increasing order. You can use any sorting algorithm. Add the 's' command in the main loop.

```
...
Command? p
Concordance
         a: 100 200
         b: 20
         c:
         d:
         e: 30
Command? s
Command? p
Concordance
         c:
         d:
         b: 20
         e: 30
         a: 100 200
Command? q
Bye!
```

Your program should now pass test cases 1-20.

Submission: Submit your file that implements the last task on Oncourse. You can resubmit as often as you want until the deadline.

- 3/8, v1.0: Removed PYNQ. Added command loop, more functions, malloc, and structs.
- 11/8 v1.1: More instructions for when using libasan.
- 5/10 v1.2: Simplified the main's index: now only used by readFile, printOriginal computes the maximum index.
- 9/10 v1.3: Show the content of the test files.
- 14/10 v1.4: Removed f from test 10.
- 20/10 v1.5: Specified max word length 10. Clarified full concordance.

Input / output test cases

Long lines have been wrapped at 70 characters for legibility. When your program output is compared to the expected output lines will not be wrapped.

Case 01

Input:

```
A
q
```

Output:

```
Command? Unknown command 'A'
Command? Bye!
```

Case 02

Input:

```
w
Apple
p
w
Pear
p
w
Apple
p
w
Orange
p
q
```

Output:

```
Command? Word? Command? Concordance
     Apple:
Command? Word? Command? Concordance
     Apple:
      Pear:
Command? Word? Command? Concordance
     Apple:
      Pear:
Command? Word? Command? Concordance
     Apple:
    Orange:
      Pear:
Command? Bye!
```

Case 03

Input:

```
p
w
dd!
w
aa?
w
bb.
w
eee
w
ccc
p
q
```

Output:

```
Command? The concordance is empty
Command? Word? Command? Word? Command? Word? Command? Word? Command?
Word? Command? Concordance
       aa?:
       bb.:
       ccc:
       dd!:
       eee:
Command? Bye!
```

Case 04

Input:

```
w
Eline
w
Vere
w
Meren
w
Koele
p
w
Kaas
w
Max
w
Havelaar
p
q
```

Output:

```
Command? Word? Command? Word? Command? Word? Command? Word? Command?
Concordance
     Eline:
     Koele:
     Meren:
      Vere:
Command? Word? Command? Word? Command? Word? Command? Concordance
     Eline:
  Havelaar:
      Kaas:
     Koele:
       Max:
     Meren:
      Vere:
Command? Bye!
```

Case 05

Input:

```
w
dddd
w
bb
w
ccc
w
a
w
ggggggg
w
hhhhhhhh
w
iiiiiiiii
w
ffffff
w
eeeee
p
q
```

Output:

```
Command? Word? Command? Word? Command? Word? Command? Word? Command?
Word? Command? Word? Command? Word? Command? Word? Command? Word?
Command? Concordance
         a:
        bb:
       ccc:
      dddd:
     eeeee:
    ffffff:
   ggggggg:
  hhhhhhhh:
 iiiiiiiii:
Command? Bye!
```

Case 06

Input:

```
w
Apple
i
Apple 2
p
w
Pear
i
Pear 1
i
Apple 3
p
i
Lychee 100
q
```

Output:

```
Command? Word? Command? Word index? Command? Concordance
     Apple: 2
Command? Word? Command? Word index? Command? Word index? Command?
Concordance
     Apple: 2 3
      Pear: 1
Command? Word index? Word Lychee not found
Command? Bye!
```

Case 07

Input:

```
w
stokkens
w
grokkens
w
scholier
w
schalker
i
stokkens 100
i
grokkens 101
i
scholier 200
i
scholier 201
i
scholier 199
i
stokkens 99
w
houtje
p
i
houtje 10
p
i
houtje 20
p
i
houtje 5
p
w
touwtje
i
touwtje 15
i
houtje 12
i
touwtje 13
p
q
```

Output:

```
Command? Word? Command? Word? Command? Word? Command? Word? Command?
Word index? Command? Word index? Command? Word index? Command? Word
index? Command? Word index? Command? Word index? Command? Word?
Command? Concordance
  grokkens: 101
    houtje:
  schalker:
  scholier: 200 201 199
  stokkens: 100 99
Command? Word index? Command? Concordance
  grokkens: 101
    houtje: 10
  schalker:
  scholier: 200 201 199
  stokkens: 100 99
Command? Word index? Command? Concordance
  grokkens: 101
    houtje: 10 20
  schalker:
  scholier: 200 201 199
  stokkens: 100 99
Command? Word index? Command? Concordance
  grokkens: 101
    houtje: 10 20 5
  schalker:
  scholier: 200 201 199
  stokkens: 100 99
Command? Word? Command? Word index? Command? Word index? Command?
Word index? Command? Concordance
  grokkens: 101
    houtje: 10 20 5 12
  schalker:
  scholier: 200 201 199
  stokkens: 100 99
   touwtje: 15 13
Command? Bye!
```

Case 08

Input:

```
p
r
test1
p
q
```

Output:

```
Command? The concordance is empty
Command? File name? Inserted 15 words
Command? Concordance
       are: 3
      file: 8
     hello: 0
      here: 14
       how: 2 5
        is: 6 9 13
        it: 10 12
     there: 1 11
      this: 7
       you: 4
Command? Bye!
```

Case 09

Input:

```
p
r
test1
p
r
test2
p
r
no-file
q
```

Output:

```
Command? The concordance is empty
Command? File name? Inserted 15 words
Command? Concordance
       are: 3
      file: 8
     hello: 0
      here: 14
       how: 2 5
        is: 6 9 13
        it: 10 12
     there: 1 11
      this: 7
       you: 4
Command? File name? Inserted 7 words
Command? Concordance
  Dead-Sea: 15
       are: 3
engineered: 19
      file: 8
     hello: 0
      here: 14
       how: 2 5
        is: 6 9 13
        it: 10 12
      like: 20
   reverse: 18
   scrolls: 16
     there: 1 11
      this: 7
     this!: 21
      were: 17
       you: 4
Command? File name? Cannot open file no-file
Command? Bye!
```

Case 10

Input:

```
r
test2
i
like 10
i
like 9
p
W
like
W
Dead-Sea
W
this!
p
W
reverse
W
scrolls
W
were
W
engineered
p
q
```

Output:

```
Command? File name? Inserted 7 words
Command? Word index? Command? Word index? Command? Concordance
  Dead-Sea: 0
engineered: 4
      like: 5 10 9
   reverse: 3
   scrolls: 1
     this!: 6
      were: 2
Command? Word? Command? Word? Command? Word? Command? Concordance
engineered: 4
   reverse: 3
   scrolls: 1
      were: 2
Command? Word? Command? Word? Command? Word? Command? Word? Command?
The concordance is empty
Command? Bye!
```

Case 11

Input:

```
w
Apple
i
Apple 3
p
w
Pear
i
Pear 1
i
Apple 3
w
Lychee
i
Lychee 90
i
Lychee 100
i
Apple 22
w
Empty
p
W
Apple
p
W
Pear
p
W
Empty
p
W
Lychee
p
q
```

Output:

```
Command? Word? Command? Word index? Command? Concordance
     Apple: 3
Command? Word? Command? Word index? Command? Word index? Command?
Word? Command? Word index? Command? Word index? Command? Word index?
Command? Word? Command? Concordance
     Apple: 3 3 22
     Empty:
    Lychee: 90 100
      Pear: 1
Command? Word? Command? Concordance
     Empty:
    Lychee: 90 100
      Pear: 1
Command? Word? Command? Concordance
     Empty:
    Lychee: 90 100
Command? Word? Command? Concordance
    Lychee: 90 100
Command? Word? Command? The concordance is empty
Command? Bye!
```

Case 12

Input:

```
w
stokkens
w
grokkens
w
scholier
w
schalker
i
stokkens 100
i
grokkens 101
i
scholier 200
i
scholier 201
i
scholier 199
i
stokkens 99
w
houtje
p
i
houtje 10
p
i
houtje 20
p
i
houtje 5
p
w
touwtje
i
touwtje 15
i
houtje 12
i
touwtje 13
p
W
grokkens
W
touwtje
p
W
scholier
p
W
schalker
W
stokkens
p
W
houtje
p
q
```

Output:

```
Command? Word? Command? Word? Command? Word? Command? Word? Command?
Word index? Command? Word index? Command? Word index? Command? Word
index? Command? Word index? Command? Word index? Command? Word?
Command? Concordance
  grokkens: 101
    houtje:
  schalker:
  scholier: 200 201 199
  stokkens: 100 99
Command? Word index? Command? Concordance
  grokkens: 101
    houtje: 10
  schalker:
  scholier: 200 201 199
  stokkens: 100 99
Command? Word index? Command? Concordance
  grokkens: 101
    houtje: 10 20
  schalker:
  scholier: 200 201 199
  stokkens: 100 99
Command? Word index? Command? Concordance
  grokkens: 101
    houtje: 10 20 5
  schalker:
  scholier: 200 201 199
  stokkens: 100 99
Command? Word? Command? Word index? Command? Word index? Command?
Word index? Command? Concordance
  grokkens: 101
    houtje: 10 20 5 12
  schalker:
  scholier: 200 201 199
  stokkens: 100 99
   touwtje: 15 13
Command? Word? Command? Word? Command? Concordance
    houtje: 10 20 5 12
  schalker:
  scholier: 200 201 199
  stokkens: 100 99
Command? Word? Command? Concordance
    houtje: 10 20 5 12
  schalker:
  stokkens: 100 99
Command? Word? Command? Word? Command? Concordance
    houtje: 10 20 5 12
Command? Word? Command? The concordance is empty
Command? Bye!
```

Case 13

Input:

```
w
houtje
W
houtje
p
w
boutje
i
boutje 10
p
i
boutje 20
p
W
boutje
i
boutje 5
w
boutje
i
boutje 22
p
W
boutje
w
snauwtje
i
snauwtje 33
i
snauwtje 44
p
W
snauwtje
p
q
```

Output:

```
Command? Word? Command? Word? Command? The concordance is empty
Command? Word? Command? Word index? Command? Concordance
    boutje: 10
Command? Word index? Command? Concordance
    boutje: 10 20
Command? Word? Command? Word index? Word boutje not found
Command? Word? Command? Word index? Command? Concordance
    boutje: 22
Command? Word? Command? Word? Command? Word index? Command? Word
index? Command? Concordance
  snauwtje: 33 44
Command? Word? Command? The concordance is empty
Command? Bye!
```

Case 14

Input:

```
w
houtje
W
houtje
p
w
boutje
i
boutje 10
p
i
boutje 20
p
W
boutje
i
boutje 5
w
boutje
i
boutje 22
p
W
boutje
w
snauwtje
i
snauwtje 33
i
snauwtje 44
p
W
snauwtje
W
snauwtje
p
q
```

Output:

```
Command? Word? Command? Word? Command? The concordance is empty
Command? Word? Command? Word index? Command? Concordance
    boutje: 10
Command? Word index? Command? Concordance
    boutje: 10 20
Command? Word? Command? Word index? Word boutje not found
Command? Word? Command? Word index? Command? Concordance
    boutje: 22
Command? Word? Command? Word? Command? Word index? Command? Word
index? Command? Concordance
  snauwtje: 33 44
Command? Word? Command? Word? Word snauwtje not found
Command? The concordance is empty
Command? Bye!
```

Case 15

Input:

```
f
10
w
Apple
i
Apple 2000
i
Apple 2003
i
Apple 1984
f
2000
f
2003
f
2020
q
```

Output:

```
Command? Index? There is no word at index 10
Command? Word? Command? Word index? Command? Word index? Command?
Word index? Command? Index? The word at index 2000 is Apple
Command? Index? The word at index 2003 is Apple
Command? Index? There is no word at index 2020
Command? Bye!
```

Case 16

Input:

```
w
Apple
i
Apple 2000
i
Apple 2003
i
Apple 1984
w
Samsung
i
Samsung
2015
w
Oppo
i
Oppo 2020
p
f
1990
f
1984
f
2000
f
2003
f
2020
q
```

Output:

```
Command? Word? Command? Word index? Command? Word index? Command?
Word index? Command? Word? Command? Word index? Command? Word?
Command? Word index? Command? Concordance
     Apple: 2000 2003 1984
      Oppo: 2020
   Samsung: 2015
Command? Index? There is no word at index 1990
Command? Index? The word at index 1984 is Apple
Command? Index? The word at index 2000 is Apple
Command? Index? The word at index 2003 is Apple
Command? Index? The word at index 2020 is Oppo
Command? Bye!
```

Case 17

Input:

```
w
Dead-Sea
w
engineered
w
this!
w
scrolls
w
like
w
were
w
reverse
p
i
this!  6
o
i
reverse 3
o
i
Dead-Sea 0
o
i
like 5
o
i
were 2
o
i
scrolls 1
o
i
engineered 4
p
o
q
```

Output:

```
Command? Word? Command? Word? Command? Word? Command? Word? Command?
Word? Command? Word? Command? Word? Command? Concordance
  Dead-Sea:
engineered:
      like:
   reverse:
   scrolls:
     this!:
      were:
Command? Word index? Command? ? ? ? ? ? ? this!
Command? Word index? Command? ? ? ? reverse ? ? this!
Command? Word index? Command? Dead-Sea ? ? reverse ? ? this!
Command? Word index? Command? Dead-Sea ? ? reverse ? like this!
Command? Word index? Command? Dead-Sea ? were reverse ? like this!
Command? Word index? Command? Dead-Sea scrolls were reverse ? like
this!
Command? Word index? Command? Concordance
  Dead-Sea: 0
engineered: 4
      like: 5
   reverse: 3
   scrolls: 1
     this!: 6
      were: 2
Command? Dead-Sea scrolls were reverse engineered like this!
Command? Bye!
```

Case 18

Input:

```
p
s
p
w
a
i
a 10
w
c
i
c 5
w
b
i
b 12
p
s
p
q
```

Output:

```
Command? The concordance is empty
Command? Command? The concordance is empty
Command? Word? Command? Word index? Command? Word? Command? Word
index? Command? Word? Command? Word index? Command? Concordance
         a: 10
         b: 12
         c: 5
Command? Command? Concordance
         c: 5
         a: 10
         b: 12
Command? Bye!
```

Case 19

Input:

```
p
s
p
w
dddd
w
bb
w
ccc
w
a
w
gggggggg
w
xxx
w
iiiiiiiii
w
ffffff
w
eeeee
i
a 10
i
bb 11
i
ccc 3
i
dddd 6
i
eeeee 99
i
ffffff 73
i
gggggggg 2
p
s
p
s
p
q
```

Output:

```
Command? The concordance is empty
Command? Command? The concordance is empty
Command? Word? Command? Word? Command? Word? Command? Word? Command?
Word? Command? Word? Command? Word? Command? Word? Command? Word?
Command? Word index? Command? Word index? Command? Word index?
Command? Word index? Command? Word index? Command? Word index?
Command? Word index? Command? Concordance
         a: 10
        bb: 11
       ccc: 3
      dddd: 6
     eeeee: 99
    ffffff: 73
   ggggggg: 2
 iiiiiiiii:
       xxx:
Command? Command? Concordance
 iiiiiiiii:
       xxx:
   ggggggg: 2
       ccc: 3
      dddd: 6
         a: 10
        bb: 11
    ffffff: 73
     eeeee: 99
Command? Command? Concordance
 iiiiiiiii:
       xxx:
   ggggggg: 2
       ccc: 3
      dddd: 6
         a: 10
        bb: 11
    ffffff: 73
     eeeee: 99
Command? Bye!
```

Case 20

Input:

```
r
test3
p
s
p
W word
W index
W add
s
p
q
```

Output:

```
Command? File name? Inserted 20 words
Command? Concordance
       add: 2 6
        at: 14
   concord: 5 19
      file: 9
      find: 12
  function: 0
     index: 7 15
  original: 16
     print: 4
      quit: 1
      read: 8
    remove: 10
      sort: 18
      text: 17
      word: 3 11 13
Command? Command? Concordance
  function: 0
      quit: 1
       add: 2 6
      word: 3 11 13
     print: 4
   concord: 5 19
     index: 7 15
      read: 8
      file: 9
    remove: 10
      find: 12
        at: 14
  original: 16
      text: 17
      sort: 18
Command? Word? Command? Word? Command? Word? Command? Command?
Concordance
  function: 0
      quit: 1
     print: 4
   concord: 5 19
      read: 8
      file: 9
    remove: 10
      find: 12
        at: 14
  original: 16
      text: 17
      sort: 18
Command? Bye!
```