# Word List
## Problem ID: wordlist

Write a program that reads a text file, containing one word in each line, into a word list and allows the user to repeatedly perform the following actions:

- `l/<line_number>`: Print the word found in line `<line_number>`.
- `s/<search_str>`: Print all words starting with `<search_str>`.
- `a/<word>`: Append the given <word> to the word list.
- `w`: Write the (potentially changed) word list back to the file.
- `q`: Quit the program.

Example actions:

- `l/10`: Prints the word in line 10.
- `a/programming`: Appends the word “`programming`” to the word list.
- `s/pro`: Prints all words starting with “`pro`”.

Implementation details:

- Your program should first ask for the filename of the file which contains the word list.
- If the file name, provided by the user, is not found, the error message “`File not found!`” should be printed. The program then exits.
- If the argument to the line (`l`) command is neither a number nor in a valid range, the error message “`Invalid line number!`” should be printed.
- If an invalid command letter is entered (i.e. something else than `l`, `s`, `a`, `w`, or `q`) or a valid command letter with a missing argument, the program simply asks user again for the next command.
- When writing the word list back to the file, you can use the `seek` function (or something similar), to move the position of the file handle to the beginning of the file. Make sure you flush the stream as well to ensure the changes are made to the file immediately. Additionally, you must write “`File written!`” after finishing writing to the file.
- The characters provided in the input can be English letters, digits and the symbols „`+-,./`“ (and newline symbols).
- Each command will either be provided with 0 or 1 parameters, meaning each line provided as input can have at most one forward slash symbol.
- As always, in this project it is important to identify subtasks and implement functions for each of the tasks.

See the attached file `test.txt` to understand the samples.

After running sample 3, the file `test.txt`, would contain the following lines:

```
a
ability
able
about
above
accept
according
account
across
act
programming
fun
```

## Scoring

| Group | Points | Constraints |
|---|---|---|
| 1 | 10 | The specified file will not exist. |
| 2 | 20 | Only valid `l` and `q` commands will be tested. |
| 3 | 20 | Only valid `s` and `q` commands will be tested. |
| 4 | 20 | Only valid `l`, `s`, `a`, and `q` commands will be tested. |
| 5 | 20 | Only valid commands will be tested. |
| 6 | 10 | No further constraints. |

**Read** **Sample Interaction 1** **Write**

```
                                Enter filename:
bla.txt
                                File not found!
```

**Read** **Sample Interaction 2** **Write**

```
                                Enter filename:
test.txt
                                Enter action:
q
```

**Read** **Sample Interaction 3** **Write**

```
                                Enter filename:
test.txt
                                Enter action:
l/ab
                                Enter action:
l/50
                                Enter action:
l/5
                                above
                                Enter action:
s
                                Enter action:
s/acc
                                accept
                                according
                                account
                                Enter action:
a
                                Enter action:
a/programming
                                Enter action:
a/fun
                                Enter action:
w
```

```
File written!
Enter action:
```

```
q
```