Assignment 3 - Software Project, Fall 2008/2009

Due: January 12, 2009. Submit in pairs.

In this exercise you will implement a program that manipulates a string using a linked list. The program will iteratively read commands from the user (standard output) and execute them.

1 User interface

The program initially creates an empty string. The user can manipulate and print the string, or replace it with a new string. The maintenance and manipulation of the string in this program is done by a linked list of characters. The program iteratively reads the following commands from the standard input, and operates accordingly.

1. `i<c1><c2>`: The character `<c2>` is inserted after the **last** occurrence of the character `<c1>` in the current string. If `<c1>` does not appear in the current string, then `<c2>` is inserted at the end. For example, suppose the current string is "12314". Then the command `i1a` will result in "1231a4". If we now apply the command `i8b` then the result is "1231a4b" (since the character '8' does not appear in "1231a4").

2. `e<c>`: Any occurrence of the character `<c>` is erased from the current list. For example, if the current string is "123a11bb1" and the command was `e1` then the resulting string is "23abb".

3. `p`: prints the current string in square brackets, followed by a new line. For example, the string "12345" is printed as `[12345]\n`. An empty string is printed as `[]\n`.

4. `d`: The string is deleted and its memory is freed. The current string is now the empty string.

5. `w<filename>`: prints the current string into a file in `<filename>`. If a file in `<filename>` already exists, it will be overwritten (i.e. replaced by a file whose content is the string). For example, suppose the current string is "123". Then the command `w/tmp/s.txt` will create the file `/tmp/s.txt` whose content is "123" (no '\0' at the end). Note that if the string is empty, the file should be empty as well (i.e. size zero).

6. `r<filename>`: a new string, whose content is read from the file `<filename>`, is created (i.e. new linked list). This string replaces the previous string.

7. `q`: The program terminates and all memory is freed.

More features:

- All the command letters (i.e. `i, e, p, d, w,r, q`) are **not** case-sensitive. Thus, the command `I11` is equivalent to to `i11`, and entering `Q` as a command is equivalent to entering `q`.

- White spaces, apart of being used as terminators for filenames (in the `r<filename>` and `w<filename>` commands), should be ignored when a new command letter is expected. Any other appearance of a white space (e.g. when a character `<c>` is expected) should be considered as error. In particular, white spaces are prohibited in the `file path` argument.
- You can limit the length of a `<filename>` argument to 300. If the user violates this limit - it can be considered as an error.
- In any case of an error, the program prints a friendly error message to the standard error and terminates. Writing to the standard error is done using the command `fprintf(stderr, "..." ...)`, which differs from the `printf` command only in the first, extra parameter `stderr`.

2 Implementation

Reading commands from standard input can be done using the function `getchar`. You may find the functions `isspace`, `islower`, and `tolower` useful in your implementation (read their man pages to learn about what they do).

2.1 Data structure

- A string should be maintained by a linked list, where each node corresponds to one character.
- In particular, an empty list should be represented by `NULL`.

2.2 Functions

The program should implement (and use) the following functions:

1. `insertToList`: The function receives two characters, **c1** and **c2**, and the current list as arguments. It operates as defined by the `i` command.
2. `eraseFromList`: The function gets a character **c** and the current list as arguments. It operates as defined by the `e` command.
3. `printList`: The function gets the current list as an argument. It operates as defined by the `p` command.
4. `deleteList`: The function gets the current list as an argument. It operates as defined by the `d` command.
5. `writeList`: The function gets a string **filename** and the current list as arguments. It operates as defined by the `w` command.
6. `readList`: The function gets a string **filename** and the current list as arguments. It operates as defined by the `r` command.

The complete prototypes of the functions above are left for you to define. However, these prototypes should satisfy the following requirement: **Every function returns 1 on success, and 0 otherwise** (e.g problems in reading from standard input, opening a file, etc.). Note: some functions may require a pointer to the list in order to modify it (i.e. a pointer to a pointer...)

2.3 File framework

The program should consists of the following files:

- `my_list.h`: contains the definition of the data structure (i.e. linked list) maintaining the string, and the prototypes of the functions presented in Section 2.2
- `my_list.c`: contains the implementation of the prototypes in `my_list.h`.
- `main.c`: contains the main function.

2.4 Compilation

Your code should be compilable and executable on the LINUX machines in the Computer Science school. The makefile is provided at the website. To compile your programs run:

```
make all
```

The name of the executable is `my_list`. To remove the object and executable files you can use the command:

```
make clean
```

2.5 Testing

You can assume that the string characters, as well as the argument characters (e.g. `<a>, <b>`) are all alphanumeric (i.e. letters and digits), and there is no need to verify it. On the course website you will find examples for input files and their expected output files. When comparing output files, the command `diff -b` can be used.

Submission

- The source files (i.e. *.c and *.h files), makefile, and "partners.txt" files for this assignment should be submitted under `~/soft-proj09/assign3`.

  Important note: As in exercise 1, make sure that you have correct permissions (755) for all the directories and files.

- Manual submission (one for each pair) should consist of printouts of source files, attached to a **printed** page with: ID, user-name, and name - of both partners.

Good Luck!