פרוייקט תוכנה - תרגיל #3

מגישים:
אריאל סטולרמן
ודים סטוטלנד

my_list.h:

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

/* Data structure definition: */

struct linked_list {
      char              d;
      struct linked_list      *next;
};

typedef struct linked_list ELEMENT;
typedef ELEMENT *LINK;

/* Functions prototypes: */

int insertToList(char c1, char c2, LINK *lst);

int eraseFromList(char c, LINK *lst);

int printList(LINK lst);

int deleteList(LINK *lst);

int writeList(char *filename, LINK lst);

int readList(char *filename, LINK *lst);
```

my_list.c:

```
#include "my_list.h"

int insertToList(char c1, char c2, LINK *lst){

	LINK tmp_lst = *lst;
	LINK tmp = (ELEMENT *)malloc(sizeof(ELEMENT));
	LINK current = NULL;
	tmp -> d = c2;

	if (tmp_lst == NULL) {		/* lst is empty */
		tmp -> d = c2;
		tmp -> next = NULL;
		tmp_lst = tmp;
		*lst = tmp_lst;		/* set original list to the only element that
contains c2 */
	} else {		/* lst is not empty, look for c2 in lst */

		while ((tmp_lst -> next) != NULL){
			if ((tmp_lst -> d) == c1) current = tmp_lst;
			tmp_lst = tmp_lst -> next;
		}
		if ((tmp_lst -> d) == c1) current = tmp_lst;

		if (current == NULL)
			tmp_lst -> next = tmp;
		else {
			tmp -> next = current -> next;
			current -> next = tmp;
		}
	}


	return 1;
}

int eraseFromList(char c, LINK *lst){

	LINK tmp_lst = *lst;
	LINK tmp = NULL;
```

```
	/* handle the beginning of the list until first d != c: */

	while ((tmp_lst != NULL)&&((tmp_lst -> d) == c)){
		tmp = tmp_lst;
		tmp_lst = tmp_lst -> next;
		free(tmp);
	}
	*lst = tmp_lst;		/* set original list to point the first element that
doesn't contain d == c */

	/* handle the rest of the list: */

	if (tmp_lst != NULL) {
		while ((tmp_lst -> next) != NULL){
			if ((tmp_lst -> next -> d) == c){
				tmp = tmp_lst -> next;
				tmp_lst -> next = tmp_lst -> next -> next;
				free(tmp);
			} else	tmp_lst = tmp_lst -> next;
		}
	}

	return 1;
}

int printList(LINK lst){

	char ch;

	if (putchar('[') != '[') return 0;

	/* printing the string: */
	while (lst != NULL){
		ch = lst -> d;
		if (putchar(ch) != ch) return 0;
		lst = lst -> next;
	}

	if (printf("]\n") < 0) return 0;

	return 1;
}
```

```
int deleteList(LINK *lst){

	LINK tmp = *lst;
	LINK current = NULL;

	while (tmp != NULL){
		current = tmp;
		tmp = tmp -> next;
		free(current);
	}

	*lst = NULL;

	return 1;
}


int writeList(char *filename, LINK lst){

	FILE *my_file = fopen(filename, "w");
	if (my_file == NULL) return 0;

	while (lst != NULL){
		if (fputc((lst -> d), my_file) == EOF) return 0;
		lst = lst -> next;
	}

	if (fclose(my_file) == EOF) return 0;

	return 1;
}


int readList(char *filename, LINK *lst){

	char ch;
	int i = 1;  /* indicator for continue read from file, at the do-while loop */
	LINK head = NULL;
	LINK current = NULL;

	FILE *my_file = fopen(filename, "r");

	deleteList(lst);  /* for freeing memory */
```

```
	if (my_file == NULL) return 0;

	ch = fgetc(my_file);

	if (ch == EOF){
		if (fclose(my_file) != 0) return 0;
	}

	/* the file contains at least one char: */

	else {
		current = (ELEMENT *)malloc(sizeof(ELEMENT));
		current -> d = ch;
		current -> next = NULL;
		head = current;

		/* to handle the rest of the chars in the file: */

		do {
			ch = fgetc(my_file);
			if (ch == EOF){
				if (fclose(my_file) != 0) return 0;
				i = 0;
			} else {
				current -> next = (ELEMENT *)malloc(sizeof(ELEMENT));
				current = current -> next;
				current -> d = ch;
				current -> next = NULL;
			}
		} while (i == 1);
	}

	*lst = head;

	return 1;
}
```

main.c:

```
#include "my_list.h"
#include <ctype.h>

int main(void){

	/* creating the empty list: */

	LINK lst = NULL;

	char ch, c1, c2;
	char filename[301];
	int i = 0;

	/* commands read loop: */

	while (((ch = getchar()) != EOF) && (i == 0)){
		if (isspace(ch) == 0) {
			if (islower(ch) == 0) ch = tolower(ch);

			switch(ch){
				case 'i':
					c1 = getchar();
					if ((isspace(c1) != 0) || (c1 == EOF)){
						fprintf(stderr, "Not a valid input for command i\n");
						i = 1;
					}
					c2 = getchar();
					if ((isspace(c2) != 0) || (c2 == EOF)){
						fprintf(stderr, "Not a valid input for command i\n");
							i = 1;
					    }
					if (insertToList(c1, c2, &lst) != 1){
						fprintf(stderr, "Error in command i\n");
							i = 1;
					    }
					break;
```

```
                        case 'e':
                              c1 = getchar();
                                    if ((isspace(c1) != 0) || (c1 == EOF)){
                                          fprintf(stderr, "Not a valid input for
command e\n");
                                          i = 1;
                                    }
                              if (eraseFromList(c1, &lst) != 1){
                                    fprintf(stderr, "Error in command e\n");
                                          i = 1;
                                    }
                                    break;
                        case 'p':
                              if (printList(lst) != 1){
                                     fprintf(stderr, "Error in command p\n");
                                          i = 1;
                                    }
                                    break;
                        case 'd':
                              if (deleteList(&lst) != 1){
                                           fprintf(stderr,   "Error   in   command
d\n");
                                          i = 1;
                                    }
                                    break;
                        case 'w':
                              c1 = '1';
                              while ((c1 != EOF) && ((c1 = getchar()) != EOF)){
                                    if (isspace(c1) != 0) c1 = EOF;
                                    else {
                                          filename[i] = c1;
                                          i++;
                                    }
                              }
                              filename[i] = '\0';
                              i = 0;

                              if (writeList(filename, lst) != 1){
                                    fprintf(stderr, "Error in command w\n");
                                          i = 1;
                                    }
                                    break;
```

```
                        case 'r':
                              c1 = '1';
                                    while ((c1 != EOF) && ((c1 = getchar()) != EOF)){
                                          if (isspace(c1) != 0) c1 = EOF;
                                          else {
                                                filename[i] = c1;
                                                i++;
                                          }
                                    }
                                    filename[i] = '\0';
                              i = 0;

                              if (readList(filename, &lst) != 1){
                                    fprintf(stderr, "Error in command r\n");
                                          i = 1;
                                    }
                              break;
                        case 'q':
                              if (deleteList(&lst) != 1)
                                          fprintf(stderr, "Error in command q\n");
                              i = 1;
                                    break;
                        default: fprintf(stderr, "Error, unknown command\n");
                  }
            }
      }

      return 0;
}
```