Εργαστήριο Διαδικαστικού Προγραμματισμού 2022-2023

Τέταρτη Εργαστηριακή Άσκηση - b

Για το δίωρο του εργαστηρίου

Ερώτημα Α: (`task1_a.c`)

Δίνεται ο κάτωθι κώδικας

```c
#include <stdio.h>
#include <string.h>
struct charact {
	char ch;
	int sec;
};
typedef struct charact Char;
Char distance(char name[50]);
void report(Char temp);

int main(void) {
	char name[50];
	Char first;
	scanf("%49s", name);

	first = distance(name);
	report(first);

	return 0;
}
Char distance(char name[50]) {
	Char temp ;

	/* Your code here! No printf or equivalent!!!
	 * (really few lines)
	 */


    return temp;
}

void report(Char t) {

	/* Your code here! No more than two lines */

	return;
}
```

Χωρίς να αφαιρέσετε οτιδήποτε, να προσθέσετε κώδικα στις συναρτήσεις `distance` και `report`, στα σημεία που υποδεικνύονται μόνο, έτσι ώστε όταν εκτελείται το πρόγραμμα, να τυπώνεται η απόσταση του πρώτου χαρακτήρα από την δεύτερη εμφάνιση του ίδιου χαρακτήρα στη λέξη, για παράδειγμα για είσοδο `"echelon"` τυπώνεται

```
e
3
```

Αν δεν υπάρχει δεύτερη φορά, θα τυπώνεται 0. Αν υπάρχει τρίτη εμφάνιση, αγνοείται.

Σημαντικό: Στη συνάρτηση `distance` να **μην** χρησιμοποιήσετε `printf` ή διαδικασίες παρόμοιας λογικής, για εκτύπωση χαρακτήρων.

Ερώτημα Β. (task1_b.c)

Δίνεται ο ακόλουθος κώδικας

```
#include <stdio.h>
#include <string.h>
struct charact {
	char ch;
	int sec;
};
typedef struct charact Char;
void letters(char name[50], Char chars[50]);
void report(Char chars[50]);

int main(void) {
	char name[50];
	Char chars[50];

	scanf("%49s", name);

	letters(name, chars);
	report(chars);

	return 0;
}

void letters(char name[50], Char chars[50]) {
	size_t i, j;
	memset(chars, 0, 50*sizeof (Char));
	/* Your code here! No printfs or equivalent
	(No more than 10 lines, are required.
	Can be done with less) */
	return;
}
void report(Char t[50]) {
	/* your code here (5-6 lines) */
	return;
}
```

Συμπληρώστε κώδικα όπως και νωρίτερα, ώστε να τυπώνεται για κάθε γράμμα του name η απόστασή του από την επόμενη εμφάνισή του ή 0, ως εξής:

```
hello
h: 0
e: 0
l: 1
l: 0
o: 0
```

Ερώτημα Γ (task1_c.c)

Δίνεται ο κάτωθι κώδικας.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define CHARS 10
char **readText(int * words) ;

int main(void) {

	char ** mytext;
	int i, words;

	mytext = readText(&words);

	for (i=0; i<words; i++)
		printf("%s\n", mytext[i]);

	return 0;
}


char **readText(int * words) {
	char **mytext = NULL;
	char *word;
	int i;

	*words = 0;
	while (scanf("%s", word=malloc(CHARS*sizeof(char))),
				strcmp(word,"TELOS")) {
		(*words) ++;
		mytext = realloc(mytext, (*words)*sizeof(char *));
	  mytext[*words-1] = word;
	}
	free(word);

	return mytext;
 }
```

Α) (task1_c_1.c) Τροποποιήστε τη συνάρτηση readText και ώστε να επιστρέφει μια μεταβλητή τύπου Text_t ο οποίος ορίζεται ως εξής

```
typedef struct text_t { char ** t; int words;} Text_t;
```

Μια μεταβλητή αυτού του τύπου περιέχει τη διεύθυνση στην οποία αρχίζει το κείμενο που εισάγεται από το πληκτρολόγιο καθώς και το πλήθος των λέξεων από τις οποίες αποτελείται. Το πρότυπο της νέας συνάρτησης θα πρέπει να είναι

```
Text_t readText(void);
```

Τροποποιήστε κατάλληλα και τη συνάρτηση `main`

Β) (`task1_c_2.c`) Χρησιμοποιήστε τη νέα συνάρτηση και τον κώδικα του ερωτήματος Β, ώστε να εκτελέσετε τα ζητούμενα του ερωτήματος Β για κάθε λέξη του κειμένου που επέστρεψε η συνάρτηση `readText`.

Γ) (`task1_c_3.c`) Τροποποιήστε περαιτέρω τη συνάρτηση `readText` ώστε να διαβάζει λέξεις από ένα εξωτερικό αρχείο κειμένου.

Μέρος Δεύτερο (παράδοση σε δύο εβδομάδες)

Ερώτημα Α (task2_a.c)

Δίνεται ο κάτωθι κώδικας. Να προσθέσετε κώδικα στα σημεία όπου ζητείται, υλοποιώντας την ίδια λειτουργία με το Ερώτημα Β του πρώτου μέρους (άσκηση στο εργαστήριο).

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct charact {
	char ch;
	int occurs;
	struct charact *next;
};
typedef struct charact Char;
typedef Char * ListofChar;
typedef Char * CharNode_ptr;
void letters(char name[50], ListofChar * chars_ptr);
void report(ListofChar chars);
Char * createnode(char ch);
int main(void) {
	char name[50];
	ListofChar chars = NULL;
	scanf("%49s", name);

	letters(name, &chars);
	report(chars);
	return 0;
}

Char * createnode(char ch) {
	CharNode_ptr newnode_ptr ;
	newnode_ptr = malloc(sizeof (Char));
	newnode_ptr -> ch = ch;
	newnode_ptr -> occurs = 0;
	newnode_ptr -> next = NULL;

	return newnode_ptr;
}

void letters(char name[50], ListofChar * lst_ptr) {
	/* add your code */
	return;
}

void report(ListofChar chars) {
	/* add your code */
	return;
}
```

Ερώτημα Β (task2_b.c) Να τροποποιήσετε το τελικό παραδοτέο της εργαστηριακής άσκησης 3, χρησιμοποιώντας δυναμικούς πίνακες, όπως στο ερώτημα Γ του πρώτου μέρους της παρούσας άσκησης, ώστε να μπορεί να χειριστεί τα αρχεία **englishWords.txt** ως λεξιλόγιο και **AlicesAdventuresInWonderland.txt** ως κείμενο, τα οποία βρίσκονται στα έγγραφα του eclass, χωρίς αλλαγή του default μεγέθους του stack.

Ερώτημα Γ (task2_c.c) Να τροποποιήσετε το τελικό παραδοτέο της εργαστηριακής άσκησης 3, χρησιμοποιώντας διασυνδεδεμένες λίστες αντί για δυναμικούς πίνακες για την αποθήκευση του λεξιλογίου και του κειμένου στη μνήμη. Βασικός τύπος για τον κόμβο της λίστας θα πρέπει να είναι ο ακόλουθος

```
typedef struct node {
      char * str;
      struct node * node ;
      } Node;
```

Κάθε μέλος str δείχνει σε σειρές διαδοχικών χαρακτήρων του κειμένου, οι οποίες χωρίζονται με τουλάχιστον ένα χαρακτήρα που επαληθεύει τη συνάρτηση isspace του ctype.h

```
Για παράδειγμα για το κείμενο
Hello, you there !
Bye
```

Το πρόγραμμα θα δημιουργεί απλά διασυνδεδεμένη λίστα τεσσάρων κόμβων, με περιεχόμενα

Hello, → you → there → ! → Bye

Ζητείται να αποφύγετε τη χρήση καθολικών (global) μεταβλητών, να χρησιμοποιήσετε πρότυπο C99, και να προσπαθήσετε να χρησιμοποιήσετε τον μεγαλύτερο αριθμό συναρτήσεων.

Να παραδώσετε αρχείο pdf στο αρχείο να **απαντήσετε αιτιολογημένα** στην ερώτηση ποια από τις υλοποιήσεις σας των Ερωτημάτων Β και Γ δεσμεύει λιγότερη μνήμη για την αποθήκευση του λεξιλογίου και του κειμένου.