Complementi di Programmazione

Esercitazione 4

Usare un main per testare il corretto funzionamento delle funzioni implementate.

Nota: Nei seguenti esercizi non è consentito utilizzare funzioni di libreria che manipolano array e stringhe.

Array

Esercizio 4.1

Scrivere una funzione

```
void random_array(int v[], int dim, int max_value)
```

che dato in input un array `v` di dimensione `dim`, e un valore intero massimo `max_value`, popoli l'array con valori casuali compresi tra zero e `max_value` (escluso).
Nota: usare la funzione `rand()` che si trova in `stdlib.h`.
reference: https://en.cppreference.com/w/c/numeric/random/rand

Esercizio 4.2

Scrivere la funzione

```
double* vec_clone(const double v[], int dim);
```

che, dato in ingresso un vettore `v` di dimensioni `dim` allochi e restituisca una copia del vettore `v`.

Esercizio 4.3

Scrivere la funzione

```
void vec_scale(double v[], int dim, double scale)
```

che dato in ingresso un vettore `v` di dimensione `dim` , modifichi `v` scalando le sue componenti di un fattore `scale`. Per "scalare" si intende moltiplicare tutte le sue componenti del valore `scale`.

Esercizio 4.4

Scrivere una funzione

```
int * numeri_unici(const int array[], int dim, int *output_dim);
```

che, dato un input un array di interi lungo `dim`, ritorni un nuovo array che non contiene ripetizioni (solo la prima occorrenza di ogni numero va mantenuta). La lunghezza dell'array ritornato deve essere salvata nell'intero puntato da `output_dim`.

Esempio

input: array = [2, -11, 0, -11, 2, 2, 1]
dim = 7
output: return value = [2, -11, 0, 1]
output_dim = 4

Stringhe

Nota: nei seguenti esercizi, tutte le funzioni che ritornano una nuova stringa devono allocare dinamicamente la memoria per la stringa.

Esercizio 4.5

Scrivere la funzione

```
void print_chars(const char s[], const int idxs[], int dim);
```

che, data in input la stringa `s` e un array di indici `idxs` con la sua dimensione `dim` , stampi a schermo i caratteri nella stringa corrispondenti agli indici. (Nota: se un valore di `idxs` non è nel range corretto, esso va ignorato).

Esercizio 4.6

Scrivere la funzione

```
char* select_chars(const char s[], const int idxs[], int dim);
```

simile alla funzione precedente, ma invece di stampare a schermo, essa restituisce una stringa composta dai caratteri corrispondenti agli indici.

Esercizio 4.7

Scrivere la funzione

```
char* invert_string(const char s[]);
```

che, data in input una stringa `s` restituisca in output la stringa `s` con i caratteri invertiti (scambiando il primo carattere con l'ultimo e così via).

Esercizio 4.8

Scrivere una funzione

```
char * mocking_spongebob(const char s[], int step);
```

che, data in input una stringa `s`, restituisce una nuova stringa in cui uno ogni `step` caratteri è maiuscolo. Nel conteggio degli step non bisogna contare gli spazi e, se il carattere è già maiuscolo, deve rimanere invariato.

Esempio:

```
step = 2
input: "Non puoi insegnare TDP con i meme"
output: "NoN pUoI iNsEgNaRe TDP cOn I mEmE"
```

Esercizio 4.9

Scrivere la funzione

```
char* capitalizer(const char s[]);
```

che, dati in input una stringa `s` , costruisca e restituisca in output una nuova stringa in cui il primo carattere di ciascuna parola nella stringa di partenza è stato reso maiuscolo. Tutti gli altri caratteri devono essere resi minuscoli.

Esempio:

input: `"i punTAtoRi Sono semPLici"`
output: `"I Puntatori Sono Semplici"`

Esercizio 4.10

Scrivere la funzione

```
char** capitalizer_strings(const char* s[], int len);
```

che, dati in input un array di stringhe `s` e il suo numero di elementi len, restituisca un nuovo array di stringhe dove tutte le stringhe di s sono state trasformate secondo la funzione `capitalizer` descritta nell'esercizio 4.9.

Esempio:

Input
s = [`"Stai imparando CDP?"`, `"i punTAtoRi Sono semPLici"`]
len = 2

output:
[`"Stai Imparando Cdp?"`, `"I Puntatori Sono Semplici"`]

File

Esercizio 4.11

Scrivere una funzione

```
char ** read_lines(const char filepath[], int *lines_n)
```

che effettua la lettura di un file di testo e ritorna un array di stringhe contenente le righe del file.
Il formato del file di testo è il seguente: la prima riga contiene solo un intero positivo, che indica il numero di righe del file (esclusa la prima). Le successive righe, ognuna terminata da un carattere di newline, sono un normale testo da leggere. Si assuma che tutte le righe nel file siano lunghe al più 80 caratteri (newline inclusi).
La lunghezza dell'array ritornato deve essere salvata nell'intero puntato da `lines_n`.
Per controllare la correttezza del programma, si faccia stampare al main le righe lette (e si rilasci correttamente la memoria).

Suggerimento:

si può usare la funzione fgets. Reference: https://en.cppreference.com/w/c/io/fgets

Attenzione: il file di testo da leggere deve contenere solo caratteri ASCII.