Compito di Programmazione I - Bioinformatica

2 settembre 2022 (tempo disponibile: 2 ore)

Esercizio 1 (31 punti)

(si consegni `vulcaniano.c`)

Un array di caratteri è detto *minuscolo* se i suoi caratteri sono tutti lettere minuscole dell'alfabeto inglese (eventualmente ripetute). Per esempio, l'array `['j','p','b','j','i','e']` è minuscolo mentre l'array `['j','(','b','j','A','e']` non è minuscolo.

Un array di caratteri è detto *vulcaniano* se è minuscolo e, inoltre, è composto da una prima parte che contiene vocali, in ordine alfabetico, e da una seconda parte contiene consonanti, sempre in ordine alfabetico. Per esempio, l'array

$$[\ \underbrace{\texttt{'e','i'}}_{\text{vocali in ordine alfabetico}}\ ,\ \underbrace{\texttt{'b','j','j','p'}}_{\text{consonanti in ordine alfabetico}}\ ]$$

è vulcaniano. Invece l'array `['e','i','B','j','j','p']` non è vulcaniano, perché non è minuscolo. Neanche l'array `['e','b','i','j','j','p']` è vulcaniano, perché la vocale `i` segue la consonante `b`. Neanche l'array `['e','i','j','j','b','p']` è vulcaniano, perché le consonanti non sono in ordine alfabetico. E neanche l'array `['i','e','b','j','j','p']` è vulcaniano, perché le vocali non sono in ordine alfabetico.

Si completino le cinque funzioni del programma `vulcaniano.c`:

```c
#include "vulcaniano.h"
// COMPLETARE CON GLI #include NECESSARI

// inizializza l'array indicato, lungo length,
// in modo che diventi un array minuscolo casuale (5 punti)
void init_random(char arr[], int length) {
  // COMPLETARE
}

// stampa l'array indicato, su una riga, senza spazi fra i caratteri,
// andando a capo alla fine (2 punti)
void print(char arr[], int length) {
  // COMPLETARE
}

// riceve un array minuscolo lungo length e ne sposta i caratteri in modo
// che l'array diventi vulcaniano (13 punti)
void ordina_vulcaniano(char arr[], int length) {
  // COMPLETARE
}

// stampa una lista di caratteri, senza spazio fra i caratteri,
// andando a capo alla fine (3 punti)
void print_list(struct element_t *l) {
  // COMPLETARE
```

```
}

// riceve un array vulcaniano lungo length e restituisce una lista
// che contiene gli stessi caratteri dell'array, nello stesso ordine,
// ma senza ripetizioni: un carattere viene inserito nella lista solo
// la prima volta che compare, mentre la seconda volta non viene
// inserito nella lista (8 punti)
struct element_t *non_ripetuti(char arr[], int length) {
  // COMPLETARE
}
```

I file `vulcaniano.h` e `main.c` sono già scritti e completi, non vanno modificati e non vanno consegnati. Se servisse, si possono aggiungere funzioni ausiliarie dentro `vulcaniano.c`.

Se tutto è corretto, un esempio di esecuzione del `main.c` potrebbe essere:

```
Inserisci la lunghezza dell'array, non negativa: 6
                          Array minuscolo casuale: jpbjie
                  Array trasformato in vulcaniano: eibjjp
Lista derivata dall'array eliminando le ripetizioni: eibjp
```

Un altro esempio di esecuzione del `main.c` potrebbe essere:

```
Inserisci la lunghezza dell'array, non negativa: 21
                          Array minuscolo casuale: elnmjkbpemuaytfawnfos
                  Array trasformato in vulcaniano: aaeeoubffjklmmnnpstwy
Lista derivata dall'array eliminando le ripetizioni: aeoubfjklmnpstwy
```