Laboratorio di programmazione e Informatica 1 - A.A. 2019-2020 -

Prova per esame on-line Sesto appello - 23 febbraio 2021

ISTRUZIONI: La prima riga di ogni programma C deve contenere il proprio nome e cognome. I programmi devono essere strutturati in funzioni e completi di commenti che spieghino il procedimento.

ATTENZIONE! Non saranno valutati programmi che non passano la fase di compilazione. Si consiglia pertanto di “mettere sotto commento” le parti di programma che danno errore in compilazione.

Tempo a disposizione: 3 ore.

ESERCIZI

Esercizio 1

Data una matrice quadrata di dimensione $m \times m$ a valori in un insieme $\{0, 1, \dots, n\}$, il *carico* di una posizione (i, j) , indicato con $c(i, j)$, è dato dalla differenza tra la somma degli elementi della riga i e la somma degli elementi della colonna j . Ad esempio nella seguente matrice:

1	2	1	1
0	0	0	1
1	1	2	0
2	0	0	0

$$c(0,0) = 1, c(1,0) = -3 \text{ e } c(3,3) = 0.$$

- Scrivere una funzione in C `void Genera(int a[DIM][DIM])` che genera una matrice di DIM righe e DIM colonne a valori interi random tra 0 e 11 tale che in ogni riga ogni elemento sia diverso dall'ultimo.
- Scrivere una funzione:

```
int CalcolaCarico(int a[][COL1], int *rmax, int *cmax, int *rmin, int *cmin)
```

che calcola e restituisce il numero di posizioni della matrice a carico nullo. Inoltre calcola le posizioni con carico massimo e con carico minimo e salva i relativi indici di riga e colonna nelle variabili `*rmax` e `*cmax` e `*rmin` e `*cmin`. (Se due posizioni hanno lo stesso valore di carico massimo/minimo, sceglierne una a piacere).

- Scrivere un programma in C che:
 1. Usando la funzione `Genera`, genera una matrice di dimensione 25x25 e stampa tale matrice sullo schermo.
 2. Utilizza la funzione `CalcolaCarico` per calcolare quanti sono gli elementi di carico nullo e per individuare le posizioni degli elementi con carico massimo e minimo rispettivamente.
 3. Stampa sullo schermo le informazioni calcolate.

Esercizio 2

In biologia molecolare, le molecole di DNA possono essere viste come stringhe sull'alfabeto dei nucleotidi A = adenina, C = citosina, G = guanina, T = timina. Ad esempio: DNA: CAGCTGATCGATGCTAGCCTG.

Scrivere un programma in linguaggio C che legge dall'utente due stringhe s_1 e s_2 corrispondenti a frammenti di DNA e verifica se s_2 può essere sovrapposta su s_1 in modo che una parte iniziale (prefisso) di s_2 coincida con una parte finale (suffisso) di s_1 . Il programma dovrà dare la lunghezza della massima sovrapposizione (0 se non si possono sovrapporre). Ad esempio, se l'utente ha inserito:

$s_1 =$ CAGCTGATCGATGCTAGCCTG

$s_2 =$ AGCCTGTTGCACCTAGA

Le due stringhe si sovrappongono come segue:

```
CAGCTGATCGATGCTAGCCTG
      AGCCTGTTGCACCTAGA
```

Il programma dovrà quindi dare in output: **La massima lunghezza di sovrapposizione e' 6.**

NOTA: il programma dovrà anche verificare la correttezza dell'input e cioè che le stringhe inserite dall'utente siano effettivamente frammenti di DNA (contengano lettere permesse).

Esercizio 3

Una successione s_2 è una sottosuccessione di s_1 se gli elementi di s_2 occorrono ordinatamente (non necessariamente in modo consecutivo) in s_1 . Ad esempio, la successione di interi $\langle 2, 6, 7 \rangle$ è una sottosuccessione di $\langle 2, 1, 3, 4, 6, 7, 9 \rangle$, ma non nella sequenza $\langle 7, 2, 1, 6, 3 \rangle$ (i valori 1, 6, 7 non sono presenti nel giusto ordine).

1. Si scriva una funzione C che, date due liste di interi S_1 e S_2 , controlli se S_2 corrisponde ad una sottosuccessione di S_1 .
2. Si scriva una funzione C che, date due liste di interi S_1 e S_2 , tali che S_2 corrisponde ad una sottosuccessione di S_1 restituisca una nuova lista S_3 corrispondente alla sottosuccessione di S_1 in cui non sono presenti gli elementi di S_2 .