testi per il laboratorio #2

Esercizi fino ai sottoprogrammi inclusi, ricorsione esclusa

Esercizio 1 - Somma k

Scrivere un programma in C (ANSI 89) che acquisisce una sequenza di al più 100 valori interi e un intero strettamente positivo k. L'acquisizione della sequenza termina quando l'utente inserisce un numero negativo o nullo, oppure quando vengono acquisiti 100 valori. Il programma visualizza 1 se la sequenza contiene almeno una coppia di valori tali che la loro somma sia k, 0 altrimenti. Dopo il valore visualizzato, mettere un `'a-capo'`. Per realizzare la soluzione si sviluppi un sottoprogramma `cercasomma` che ricevuto in ingresso k, l'array contenente i dati e qualsiasi altro parametro ritenuto strettamente necessario, restituisce 1 o 0 nel caso trovi i due valori la cui somma è k.

Esercizio 2 - Solo in ordine

Scrivere un programma in C (ANSI 89) che acquisisce una sequenza di al più 20 valori interi, chiedendo all'utente inizialmente quanti valori vorrà fornire, `num`. Il programma acquisisce `num` valori e memorizza in una opportuna struttura dati la sequenza di valori i cui elementi sono strettamente crescenti, trascurando i valori che risultano non essere ordinati. Al termine dell'acquisizione il programma visualizza la lunghezza della sequenza, seguita, su una nuova riga, dalla sequenza stessa. L'utente inserirà sempre un numero di valori coerente con la richiesta.

Avvalersi di due sottoprogrammi: `fillarrord` e `viewarr`: il primo acquisisce i dati e memorizza quelli ritenuti validi, il secondo visualizza il contenuto di un array.

Esempio:

```
input:
6
1 1 6 2 5 7
output:
3
1 6 7
```

Esercizio 3 - Album musicali

Si vuole realizzare un programma in C (ANSI 89) per la gestione di un archivio di album musicali (al massimo 100). Ogni album è caratterizzato da un titolo ed un autore (entrambe stringhe di al massimo 30 caratteri), un anno di pubblicazione, il

numero di tracce, la durata complessiva (in termini di ore, minuti, secondi); infine, si vuole memorizzare anche il prezzo dell'album (un valore float).

Definire un tipo di dato per rappresentare l'archivio di album. In seguito scrivere un programma che chiede all'utente prima il numero di album da inserire e poi i dati di ciascun album (assumendo che l'utente inserisca correttamente i dati). Il programma visualizza i dati dell'album di durata massima; se sono presenti più album di stessa durata massima il programma visualizzerà i dati del primo di essi (VARIANTE: il programma visualizza i dati di tutti gli album di durata massima, se ne sono stati trovati più di uno).

In seguito il programma chiede il nome di un autore e visualizza i titoli di tutti gli album pubblicati da questo. Infine il programma visualizza l'autore che ha pubblicato più album; se più autori autori hanno lo stesso numero massimo di album, si visualizzi il primo.

Esercizio 4 - Mix di due array ordinati

Scrivere un sottoprogramma `mixarr` in C (ANSI 89) che, ricevuti come parametri tre array, `a1`, `a2` e `a3`, e qualsiasi altro parametro ritenuto strettamente necessario, salva in `a3` una copia dei valori contenuti in `a1` ed `a2` ordinandoli in ordine crescente e senza ripetizioni e restituisce il numero di elementi effettivamente scritti in `a3`. Nella realizzazione del sottoprogramma è necessario tener conto dei seguenti aspetti:

- il sottoprogramma può modificare i dati contenuti negli array di partenza;
- si assuma che `a3` abbia abbastanza spazio per memorizzare la sequenza risultante.

Nello sviluppo della soluzione realizzare un sottoprogramma `sortarr` che ricevuti in ingresso un array, un intero `updown` e qualsiasi altro parametro ritenuto strettamente necessario, ordina il contenuto dell'array in senso crescente se `updown` vale 1, in senso decrescente se vale -1. In caso `updown` contenga un valore diverso il sottoprogramma non effettua alcuna modifica. Sviluppare infine un programma che chiede all'utente i dati per popolare due array da 20 elementi, invoca il sottoprogramma sopra definito e visualizza il risultato (la visualizzazione non viene fatta nel sottoprogramma `mixarr`).

Esercizio 5 - Sottostringa distinta

Scrivere un programma in C (ANSI 89) che acquisita una stringa di al più 30 caratteri, individui la sottostringa più lunga in essa contenuta, senza caratteri ripetuti. Il programma visualizza la lunghezza di tale sottostringa, seguita da un carattere `a-capo`.

Realizzare la propria soluzione organizzandola in sottoprogrammi, come ritenuto più opportuno.

Esercizio 6 - Numeri perfetti, abbondanti e difettivi

Scrivere un programma in C (ANSI 89) che riceve in ingresso un array di 10 valori numeri interi. Costruire un nuovo array dove in corrispondenza di ogni elemento dell'array iniziale si mette:

- 0 se l'elemento `i` è non strettamente positivo,
- 1 se l'elemento `i` è un numero *perfetto*,
- 2 se l'elemento `i` è *abbondante*,
- 3 se l'elemento `i` è *difettivo*

Per definizione un numero è *perfetto* se corrisponde alla somma dei suoi divisori, escluso se stesso; *abbondante* se è minore della somma dei suoi divisori, escluso se stesso; altrimenti è *difettivo*. Ad esempio: 6 e 28 sono *perfetti*, 12 e 18 sono *abbondanti*, mentre 8 e 10 sono *difettivi*.

Il programma visualizza i valori inseriti, su una nuova riga la caratteristica del numero, ed infine un istogramma verticale che rappresenta la quantità di numeri *perfetti*, *difettivi* e *abbondanti* contenuti nell'array iniziale. Realizzare la propria soluzione organizzandola in sottoprogrammi, come ritenuto più opportuno.

Esempio:

```
input:  5 28 -1 0 6 10 18 -8 1 5
output:
5 28 -1  0  6 10 18 -8  1  5
3  1  0  0  1  3  2  0  3  3
```

```
  *
  *
* *
***
```

Esercizio articolato - k-mer

Un k-mer è una sequenza di k caratteri. Il sottoprogramma `conta` calcola e restituisce il numero di volte in cui un k-mer `kmer` compare nella sequenza (di DNA) `seq`. Per esempio, il k-mer `ACTAT` compare 3 volte nella sequenza di DNA `ACAACTATGCATACTATCGGGAACTATCCT`. Si noti che il k-mer `ATA` compare 3 volte (non 2) nella sequenza `CGATATATCCATAG`, poiché ci sono due sottosequenze parzialmente sovrapposte. Si sviluppi il sottoprogramma `conta`, il cui prototipo (con anche il nome dei parametri formali) è:

```c
int conta(char seq[], char kmer[]);
```

In alcuni contesti, non è necessario trovare la sottosequenza esatta, ma è sufficiente una approssimata, ossia che differisce da quella esatta di `d` elementi. La differenza tra due sequenze di ugual dimensione viene misurata in termini di *distanza di Hamming* ossia il numero di elementi diversi in posizione corrispondente nelle due sequenze. Per esempio, le sequenze `ATA` e `ATC` hanno una distanza pari a 1 (questa nozione di distanza di applica a sequenze di qualsiasi tipo). Si realizzi un sottoprogramma `distanza` che ricevute in ingresso due stringhe calcoli e restituisca la distanza di Hamming. Nella realizzazione si tenga presente che nell'impiego del sottoprogramma le due stringhe potrebbero avere lunghezze diverse.

```
int distanza(char seq1[], char seq2[]);
```

Si realizzi un sottoprogramma `contad` che conta e restituisce il numero di volte in cui un k-mer `kmer` compare nella sequenza `seq`, accettando anche delle forme approssimate purché a distanza minore o uguale a `d`. La valutazione deve tenere presente che l'intero k-mer deve essere contenuto nella sequenza, per cui: `ATA` non è contenuto in modo approssimato (con `d` uguale a 1) nella sequenza `CCAT`.

```
int contad(char seq[], char kmer[], int d);
```

Sviluppare un programma in C (ANSI 89) che acquisiti in ingresso una sequenza di DNA di al più 100 caratteri, il k-mer (di al più 10 caratteri) e il livello di approssimazione accettabile, calcola e visualizza (mediante i sottoprogrammi prima specificati) il numero di k-mer esatti trovati nella sequenza seguito dal numero di quelli approssimati.

Esempi:

```
input: ACAACTATGCATACTATCGGGAACTATCCT ACTAT 3
output: 3 5

input: ACAACTATGCATACTATCGGGAACTATCCT ACTAT 0
output: 3 0

input: ACGTTGCATGTCGCATGATGCATGAGAGCT GCAT 1
output: 3 0

input: ACGTTGCATGTCGCATGATGCATGAGAGCT GCAT 2
output: 3 2
```