Corso di Fondamenti di Programmazione - canali A-D e E-O
aa. 2008/2009

Homework 3, consegnare entro le 24:00 del 09/12/2008

Esercizio 1 [**Somma prefissi**]. Scrivere una funzione RICORSIVA in linguaggio C, con prototipo

```
void somma_prefissi( long [], int );
```

in grado di leggere una sequenza di lunghezza indefinita (comunque minore di 100 elementi) di interi positivi che terminino con il valore -1 come sentinella. Si deve salvare nel vettore in input nella cella i il valore della somma parziale dei primi $i + 1$ elementi. La funzione deve anche salvare il valore -1 come sentinella nell'array.

Scrivere quindi un programma completo che dichiari le variabili opportunamente e stampi i valori del vettore calcolati dalla funzione `somma_prefissi`, separati da UNO spazio o da UN carattere di nuova riga. All'interno della funzione `somma_prefissi` vengono calcolate le somme e salvate nel vettore mentre all'interno della funzione `main` tale vettore è stampato come descritto sopra (tutti gli elementi separati da UNO spazio o da UN carattere di nuova riga).

Esempio: Se l'input è

```
1  2  3  4  5  6 10 12 -1
```

il programma deve stampare

```
1  3  6 10 15 21 31 43
```

Esercizio 2 [**Somma suffissi**]. Sostituire la funzione `somma_prefissi` dell'esercizio precedente con una funzione RICORSIVA

```
long somma_suffissi( long [], int );
```

che legga una sequenza analogamente alla funzione `somma_prefissi` e salvi nella cella del vettore i la somma dei valori successivi al valore i-esimo e precedenti al valore -1.

Esempio: Se l'input è

```
1  2  3  4  5  6 10 12 -1
```

il programma deve stampare

```
43 42 40 37 33 28 22 12
```

Esercizio 3 [**Occorrenze semi-ordinate**]. Scrivere un programma in linguaggio C che legga un un intero k e un numero imprecisato di valori interi positivi, terminati dal valore -1, salvandoli in un vettore di dimensione 100 (il valore -1 deve essere memorizzato nel vettore come terminatore).
Implementare quindi una funzione RICORSIVA

```
void occorrenze ( int v [], int k, int i, int * occ_min, int * occ_max )
```

dove k corrisponde al primo valore letto, v all vettore in cui sono stati memorizzati i valori in input, i è un indice di posizione nel vettore v, mentre occ_min e occ_max conterranno gli indirizzi delle variabili in cui salvare rispettivamente le occorrenze del valore k al punto 1 e 2 di seguito descritti:

1. occ_min: quante volte il numero k compare immediatamente dopo un numero minore di k nella sequenza (se il primo numero ricevuto è uguale a k la sua occorrenza va contata);

2. occ_max: quante volte il numero k compare immediatamente prima di un numero maggiore di k nella sequenza (se il valore che precede -1 è uguale a k la sua occorrenza va contata).

La funzione `main` del programma deve quindi stampare i valori di occ_min e occ_max calcolati dalla funzione occorrenze, separati da UNO spazio o UN carattere di nuova linea.
ATTENZIONE: non è consentito stampare all'interno della funzione occorrenze e le variabili occ_min ed occ_max devono essere inizializzate all'interno della funzione `occorrenze` (in questo modo il valore calcolato è corretto a prescindere dal valore che esse assumono all'interno della funzione main)

Esempio: Se l'input è

```
9 9 7 23 8 9 9 13 9 8 5 -1
```

il programma deve stampare

```
2
1
```

infatti il primo valore letto in input (k) è 9. Le occorrenze da contare al punto 1 dell'esercizio sono quindi 2 (la prima e quella preceduta dal valore 8), mentre l'unica occorrenza da contare per il punto 2 dell'esercizio è quella seguita dal valore 13.