Compito di Programmazione - Bioinformatica

19 giugno 2024 (tempo disponibile: 2 ore)

Esercizio 1 (22 punti, **si consegni `dipendenti.c` opportunamente modificato**)

Si costruisca una lista concatenata di dipendenti caratterizzati ciascuno da codice fiscale e stipendio. Il codice fiscale e' una stringa alfanumerica formata da 16 caratteri. Esempio: CZZVTR**81**S**51**D086N. I primi due caratteri in grassetto rappresentano l'anno di nascita, 81 per il codice fiscale nell'esempio. Il secondo gruppo di caratteri in grassetto invece indica il giorno di nascita e il genere. In particolare il secondo gruppo di caratteri è pari al giorno di nascita per un uomo, al giorno di nascita più 40 per una donna. Ad esempio 51 indica una donna nata giorno 11; 10 un uomo nato giorno 10.

Si completi il file dipendenti.c di seguito riportato:

La funzione main e' già scritta e completa, non va modificata. Si possono (e si suggerisce di) aggiungere funzioni ausiliarie dentro `dipendenti.c`.

```c
#include <stdio.h>
struct entry {
    //da completare
};

struct entry *construct_entry(char fiscal_code[], float income,
  struct entry *next) {
  //da completare
}

int list_length(struct entry *l) {
  //da completare
}
float list_sum(struct entry *l) {
  //da completare
}

void print_list(struct entry *l) {
  //da completare
}

void destroy_list(struct entry *l) {
  //da completare
}

struct entry *senior_women_manager(struct entry *l, float avg_income) {
  //da completare
}

int main(void) {
   struct entry *l = NULL;
      l = construct_entry( "czzvtr81s51d086n", 1300.0, construct_entry( "
         czzvtr61s51d086n", 3300, construct_entry( "vnrmzz93s42d086n",
         1100.0, construct_entry( "abclmn00s71d086n", 3000.0,
         construct_entry( "mrtmtt93d48f205j", 3500.0, construct_entry( "
         rmqxss61s01d086n", 2500.0, l))))));
```

```
    printf("La lista e' lunga %d\n", list_length(l));
    printf("I dipendenti sono:\n");
    print_list(l);

    float avg_income= list_length(l)!=0? list_sum(l)/list_length(l): 0;
    printf("La media degli elementi e' %2.f\n", avg_income);
    printf("I dipendenti senior e con stipendio maggiore della media sono:\n"
       );
     struct entry * swm = senior_women_manager(l, avg_income);
     print_list(swm);
     destroy_list(l);
     destroy_list(swm);
   return 0;
}
```

In particolare si completino le funzioni:

- construct_entry che crea e inizializza una nodo della lista dipendenti;
- list_length che calcola la lunghezza della lista.
- list_sum che prende in input una lista e restituisce la somma degli stipendi di tutti i dipendenti.
- senior_women_manager che restituisce una lista, a partire da una lista data, che contiene solo i codici fiscali e gli stipendi delle donne che sono nate prima del 94 e che hanno degli stipendi sopra la media.

Le ultime tre funzioni indicate vanno implementate in modo **ricorsivo**. Se tutto è corretto, un esempio di esecuzione di `dipendenti.c` con la lista come inizializzata nel main è:

```
La lista e' lunga 6
I dipendenti sono:
czzvtr81s51d086n: 1300
czzvtr61s51d086n: 3300
vnrmzz93s42d086n: 1100
abclmn00s71d086n: 3000
mrtmtt93d48f205j: 3500
rmqxss61s01d086n: 2500

La media degli elementi e' 2450
I dipendenti senior e con stipendio maggiore della media sono:
czzvtr61s51d086n: 3300
mrtmtt93d48f205j: 3500
```

Esercizio 2 (8 punti, **si consegni sottostringa.c opportunamente modificato**)

Data una stringa s1 e una stringa s2, vogliamo verificare se s2 e' sottostringa di s1. Si modifichi il seguente file `sottostringa.c` completando opportunamente la funzione isSubstring:

```c
#include <stdio.h>

int isSubstring(char*b, char *a){
  //da completare
}
int main(void){
  printf("%s%s e' sottostringa di %s\n", "am", isSubstring( "am","amici")
     ==0? " non": "", "amici");
  printf("%s%s e' sottostringa di %s\n", "pia", isSubstring("pia", "amici")
     ==0? " non": "", "amici");
  printf("%s%s e' sottostringa di %s\n", "riva", isSubstring("riva", "
     arrivare")==0? "  non": "", "arrivare");
  return 0;
}
```

L'implementazione della funzione deve utilizzare **i puntatori e l'aritmetica dei puntatori** per accedere alle stringhe.

La funzione main non va modificata.

Se tutto è corretto, l'esecuzione di `sottostringa.c` stamperà:

```
am e' sottostringa di amici
pia non e' sottostringa di amici
riva e' sottostringa di arrivare
```