Seconda prova parziale di Programmazione I

23 giugno 2014 (tempo disponibile: 2 ore)

Esercizio 1 (12 punti)

Il coefficiente binomiale degli interi non negativi n e k è definito come

$$\binom{n}{k} = \frac{n!}{k! \cdot (n-k)!}$$

Si definisca una funzione

`char *stringa_binomiale(int n, int k)`

che restituisce una nuova stringa del tipo `"n su k fa b"` dove n e k sono i valori dei parametri della funzione e b è il valore di $\binom{n}{k}$. Si assuma che per scrivere un intero siano sempre sufficienti 10 caratteri.

Se tutto è corretto, un'esecuzione del seguente programma:

```
int main(void) {
  int n, k;
  char *s;
  printf("n: " ); scanf("%i", &n);
  printf("k: " ); scanf("%i", &k);
  printf("%s\n", s = stringa_binomiale(n, k));
  free(s);
  return 0;
}
```

è la seguente:

```
n: 8
k: 3
8 su 3 fa 56
```

Esercizio 2 (9 punti)

Si considerino le liste di interi come viste a lezione. Si definisca una funzione **ricorsiva**

`int sempre_diversi(struct list *this)`

che restituisce *vero* se e solo se nella lista `this` due elementi contigui sono sempre diversi.

Esercizio 3 (11 punti)

Si consideri il seguente file `vocabolario.h` che definisce la struttura e le funzioni per un vocabolario italiano/inglese che può contenere fino a 100 voci:

```
#ifndef VOCABOLARIO_H
#define VOCABOLARIO_H

struct voce {     // una singola voce italiano/inglese
  char *italiano;
  char *inglese;
};

#define SIZE 100

struct vocabolario {  // il vero e proprio vocabolario
```

```
  struct voce voci[SIZE];  // le voci contenute nel vocabolario, non tutte usate
  int voci_usate; // quante voci del vocabolario sono gia' state usate
};

struct vocabolario *construct_vocabolario(); // costruisce un vocabolario vuoto
void destroy_vocabolario(struct vocabolario *this); // dealloca un vocabolario
void print_vocabolario(struct vocabolario *this);  // stampa sul video un vocabolario
void add_in_vocabolario(struct vocabolario *this, char *italiano, char *inglese);  // aggiunge una voce al vocabolario
char *cerca_dallitaliano(struct vocabolario *this, char *italiano);  // cerca una traduzione dall'italiano
char *cerca_dallinglese(struct vocabolario *this, char *inglese);  // cerca una traduzione dall'inglese

#endif
```

Si completi la seguente implementazione `vocabolario.c`:

```
#include <stdlib.h>
#include <stdio.h>
#include "vocabolario.h"

struct vocabolario *construct_vocabolario() { COMPLETATE }
void destroy_vocabolario(struct vocabolario *this) { COMPLETATE }

void print_vocabolario(struct vocabolario *this) {
  int pos;

  printf("%-20s | %-20s\n", "italiano", "inglese");
  printf("%-20s | %-20s\n", "--------", "-------");

  for (pos = 0; pos < this->voci_usate; pos++)
      printf("%-20s | %-20s\n", this->voci[pos].italiano, this->voci[pos].inglese);
}

void add_in_vocabolario(struct vocabolario *this, char *italiano, char *inglese) { COMPLETATE }
char *cerca_dallitaliano(struct vocabolario *this, char *italiano) { COMPLETATE }
char *cerca_dallinglese(struct vocabolario *this, char *inglese) { COMPLETATE }
```

Se tutto è corretto, l'esecuzione del seguente programma:

```
#include <stdio.h>
#include "vocabolario.h"

int main(void) {
  struct vocabolario *v = construct_vocabolario();
  add_in_vocabolario(v, "cane", "dog");
  add_in_vocabolario(v, "topo", "mouse");
  add_in_vocabolario(v, "casa", "house");
  add_in_vocabolario(v, "simpatico", "nice");
  add_in_vocabolario(v, "correre", "to run");

  print_vocabolario(v);

  printf("La traduzione in inglese di 'simpatico' e' %s\n", cerca_dallitaliano(v, "simpatico"));
  printf("La traduzione in italiano di 'house' e' %s\n", cerca_dallinglese(v, "house"));

  destroy_vocabolario(v);

  return 0;
}
```

dovrà stampare:

```
italiano             | inglese
--------             | -------
cane                 | dog
topo                 | mouse
casa                 | house
simpatico            | nice
correre              | to run
La traduzione in inglese di 'simpatico' e' nice
La traduzione in italiano di 'house' e' casa
```