Informatica Generale
Homework P/2018: Naturalmente, naturali

Ivano Salvo – Sapienza Università di Roma – 26/3/2018

Esercizio 1 Scrivere una funzione C di prototipo:

```
int kthDigitB(int n, int k, int b)
   /* PREC: n>=0, k>=0, b>=2
    * POST: se n = c_m*b^m+...+c_k*b^k+..,+c_0*b^0
    * (c_m != 0) se k<=m torna c_k altrimenti torna -1
    */
```

che restituisce la k-esima cifra nella rappresentazione in base b di n. Le cifre sono numerate dimodochè $n = c_0 b^0 + c_1 b^1 + \ldots + c_m b^m = \sum_{i=0}^{\infty} c_i b^i$. Quest'ultima formula ci ricorda che per $k > m = \lfloor \log_b n \rfloor$, potremmo stipulare che $c_m = 0$. Tuttavia, il fatto che $k > m$ va segnalato restituendo -1.

Osservazione Ovviamente 'cifra' va inteso in senso lato: se b fosse 16 le cifre sarebbero numeri interi compresi tra 0 e 15.

Esercizio 2 Scrivere una funzione C di prototipo:

```
int sumTwoSquares(int n, int *x, int *y)
   /* PREC: n>=0
    * POST: ritorna 1 e (*x)^2+(*y)^2=n && *x>=*y
    * ritorna 0 se *x, *y non esistono
    */
```

che preso in ingresso un numero n restituisce 1 se n può essere scritto come somma di due quadrati (di numeri interi) e 0 altrimenti. Quando restituisce 1 inoltre, carica nelle variabili `*x` ed `*y` due valori per cui sia soddisfatta la relazione $n = x^2 + y^2$ e $x \geq y$.

Esempi: Preso in input 3, 5 oppure 6 il programma deve tornare 0.

Se l'input fosse un quadrato, $n = 1, 4, 9, 16, 25, \ldots$ il programma deve rispondere 1 e può sempre rispondere la coppia $x = \sqrt{n}$ e $y = 0$.

Tuttavia la soluzione non è necessariamente unica. Ad esempio, nel caso del 25, ad esempio, anche la coppia $x = 4, y = 3$ è una risposta valida.

Esercizio 3 Nello stato di Fibolandia hanno corso legale i Fiorinacci, moneta i cui tagli sono numeri di Fibonacci, e cioè 1, 2, 3, 5, 8, 13, 21, 34, 55, ... (osservate che a differenza della serie di Fibonacci, c'è ovviamente un'unica moneta di taglio 1). I commercianti di Fibolandia sono tutti matematici buontemponi e, dovendo dare di resto n Fiorinacci, consegnando le monete, pronunciano sempre il numero di tutti i possibili *insiemi* di tagli di monete che danno n come somma.

Esempio: Se il resto da dare fossero 9 Fiorinacci, viene pronunciato il numero 17 in quanto i possibili modi di dare il resto sono i seguenti (attenzione quindi che *non* vengono ricontati gruppi di monete che differiscono solo per l'ordine, ad esempio $\{5, 2, 1, 1\}$ e $\{1, 2, 5, 1\}$):

$$\{8,1\},\{5,3,1\},\{5,2,2\},\{5,2,1,1\},\{5,1,1,1,1\},\{3,3,3\},\{3,3,2,1\},\{3,3,1,1,1\},$$
$$\{3,2,2,2\},\{3,2,2,1,1\},\{3,2,1,1,1,1\},\{3,1,1,1,1,1,1\},\{2,2,2,2,1\},$$
$$\{2,2,2,1,1,1\},\{2,2,1,1,1,1,1\},\{2,1,1,1,1,1,1,1\},\{1,1,1,1,1,1,1,1,1\}$$

Scrivere una funzione C di prototipo:

```
int hereYouAre(int n)
   /* PREC: n>=0 */
```

che restituisce come risultato il numero pronunciato dai commercianti di Fibolandia quando danno resto n (se n fosse 0, il commerciante risponde 0).

Suggerimenti e Osservazioni: Questo esercizio non necessita di memorizzare i numeri di fibonacci: avendo una coppia di numeri consecutivi di Fibonacci, potete infatti calcolare alla bisogna sia i successivi che i precedenti.

Questo esercizio potrebbe apparire *molto* impegnativo, ma cercando con cura troverete problemi simili risolti.

Note Pratiche

Dovete consegnare semplicemente la funzione richiesta, **rispettando il prototipo**. Il vostro file può contenere eventuali altre funzioni ausiliarie. Non dovete lasciare **nessuna** istruzione di input/output nel codice. Potete verificare se il vostro programma funziona, utilizzando i files `main-P-1.c`, `main-P-2.c`, e `main-P-3.c` forniti nella pagina degli Homework.

Potete chiamare come volete il file `.c` contenenenti la funzione richiesta (ed eventuali funzioni ausiliarie): il sistema li rinominerà automaticamente usando il vostro twiki name. Se il vostro twiki name fosse `LilyEvans`, essi verranno chiamati `LilyEvans.1.c`, `LilyEvans.2.c`, e `LilyEvans.3.c`. Il comando utilizzato per la compilazione sarà:

```
gcc -std=c99 main-P-1.c LilyEvans.1.c
gcc -std=c99 main-P-2.c LilyEvans.2.c
gcc -std=c99 main-P-3.c LilyEvans.3.c
```

Osservate che i files dovrebbero poter essere compilati separatamente con l'opzione `-c` che traduce in linguaggio macchina, ma senza generare un eseguibile (`gcc -c -std=c99 LilyEvans.1.c` e `gcc -c -std=c99 main-P-1.c` dovrebbero generare con successo un file chiamato rispettivamente `LilyEvans.1.o` e `main-P-1.o`.