Informatica Generale
Homework 1/2020:
The best of '10s Homeworks.

Ivano Salvo – Sapienza Università di Roma – 18/4/2020

Esercizio 1 (2019) (Altamente Composti) Usare la logica del crivello di Eratostene (vedi *Dispensa* **D4**, **2.4**) per scrivere una funzione (vedi riquadro) che *alloca* un vettore di k elementi (che chiameremo d) e lo restituisce in output caricato mettendo in $d[i]$ il numero di divisori (non necessariamente primi) di i (questa è la sequenza A000005 dell'*On-line Encyclopedia of Integer Sequecens*). La funzione inoltre carica in ac il primo indice del massimo del vettore, cioè il più grande numero altamente composto minore di k.

```
int* maxAltamenteComposto(int k, int* ac)
   /* PREC: k>0, POST: torna un vettore d lungo k
    * tale che d[i]=numero divisori di i>0 e carica in *ac
    * il maggiore numero altamente composto < k.
    */
```

Esempio: Se k fosse 31, il vettore risultante sarebbe:

$$\#, 1, 2, 2, 3, 2, 4, 2, 4, 3, 4, 2, 6, 2, 4, 4, 5, 2, 6, 2, 6, 4, 4, 2, 8, 3, 4, 4, 6, 2, 8$$

e la funzione caricherebbe in `*ac` il numero 24 che è il primo numero con 8 divisori (# significa che non è rilevante il valore del vettore in posizione 0 ☺).

Esercizio 2 (2014) (Successione di Ulam) Consideriamo la successione di numeri naturali, nota come successione di Ulam, definita come segue: $u_0 = 1, u_1 = 2$ e u_n $(n > 1)$ è il *minimo* numero naturale che si può scrivere in *modo unico* come somma di due precedenti numeri della successione. La successione sopra definita comincia con $1, 2, 3, 4, 6, 8, 11, 13, 16, 18, 26, 28, \ldots$.

Per capire meglio la definizione, osservate che u_4 non può essere 5, in quanto $5 = 2+3 = u_1+u_2$, ma anche $5 = 1+4 = u_0+u_3$. Viceversa solo $u_1+u_3 = 2+4$ danno come somma 6.

Scrivere una funzione C di prototipo:

```
int ulam(int n)
   /* PREC: n>=0, POST: torna l'n-esimo numero di Ulam */
```

che preso in input un numero naturale n ritorna u_n. Ad esempio, se l'input fosser 4, la funzione torna 6. Se l'input fosse 11, la funzione torna 28.

Osservazioni: Potete seguire molte strade. Può essere utile allocare un vettore `u` con $n+1$ posizioni (indicizzate da 0 a n), caricate i valori iniziali in `u[0]` e `u[1]` e calcolatevi iterativamente tutta la successione partendo da `u[2]` fino a `u[n]`. Per facilitare la ricerca del $k+1$-esimo elemento u_{k+1} (una volta noti $u_0, u_1, \ldots, u_k$) osservate che necessariamente si ha che $u_k + 1 \leq u_{k+1} \leq u_{k-1} + u_k$. Infatti, se ci fosse qualche numero minore di u_k che si scrive in modo unico come somma di due precedenti, sarebbe già stato inserito nella successione. Inoltre, essendo la successione strettamente crescente, è ovvio che $u_{k-1} + u_k$ si scrive in modo unico come somma di due precedenti (in quanto questa somma è strettamente maggiore di ogni altra somma di due precedenti) e ciò, tra l'altro, dimostra che la successione è infinita (e la ricerca di u_{k+1} termina sempre).

Esercizio 3 (2015) (La sorgente d'acqua) Una matrice q di interi di dimensioni $m \times n$ rappresenta le quote di un rilievo topografico. Nelle coordinate x, y è presente una sorgente d'acqua. Sapendo che l'acqua scende (ma non sale) in tutte le direzioni (verticale, orizzontale e diagonale) verso punti con quota minore o uguale, determinare i punti che saranno bagnati dall'acqua. Dovete quindi scrivere una funzione di prototipo:

```
char** bagnatiAsciutti(int** q, int m, int n, int x, int y, int* b)
   /* PREC: q matrice m x n, 0<=x<m, 0<=y<n,
    *        [x,y] posizione della sorgente
    */
```

dove q è una matrice $m \times n$ di interi, x ed y rappresentano le coordinate della sorgente. La funzione deve allocare una matrice $m \times n$ di caratteri e

caricarla ponendo il carattere 's' (=sorgente) in posizione x, y, il carattere 'b' (=bagnato) in tutti i punti raggiunti dall'acqua, e il carattere 'a' (=asciutto) in tutti i punti non raggiunti dall'acqua. Deve inoltre caricare nella variabile `*b` il numero dei punti bagnati dall'acqua (sorgente inclusa).

ESEMPIO: avendo in input la matrice sotto a sinistra, (e quindi $m = 3$ ed $n = 5$), le coordinate della sorgente $x = 0, y = 2$, la funzione alloca e ritorna la matrice di caratteri sotto a destra. Carica inoltre nella variabile `a` il valore 8, corrispondente al numero di punti raggiunti dall'acqua (compresa la sorgente).

```
8 1 7 4 6
1 2 9 8 3
7 6 1 9 2
```

```
a b s b a
b b a a b
a a b a b
```

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-2020-1-1.c`, `main-2020-1-2.c`, e `main-2020-1-3.c` forniti nella pagina degli Homework.

Potete chiamare come volete il file `.c` contenenti 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-2020-1-1.c LilyEvans.1.c
gcc -std=c99 main-2020-1-2.c LilyEvans.2.c
gcc -std=c99 main-2020-1-3.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-2020-P-1.c` dovrebbero generare con successo un file chiamato rispettivamente `LilyEvans.1.o` e `main-2020-P-1.o`.