APPELLO SCRITTO DI PROGRAMMAZIONE 1
CORSO DI LAUREA IN MATEMATICA
UNIVERSITÀ DEGLI STUDI DI MILANO

2014–2015

17.XI.2015

VINCENZO MARRA

Indice

Avvertenza. Il tema d'esame richiede lo sviluppo di un singolo programma che permetta di eseguire delle semplici operazioni di conversione di base su numeri naturali. Svilupperete delle funzioni ausiliarie nel risolvere gli esercizi. Integrerete poi tutte le funzioni ausiliarie in un unico programma finale, che consisterà in un unico file sorgente. (Il tema d'esame richiede esplicitamente l'implementazione di alcune funzioni ausiliarie. È comunque sempre ammissibile aggiungere altre funzioni ausiliarie qualora lo riteniate opportuno per una migliore implementazione della vostra soluzione.) Si raccomanda di leggere interamente il tema d'esame con attenzione prima di cominciare a scrivere le soluzioni dei singoli esercizi, in modo da avere chiaro l'obiettivo finale.

Esercizio 1

Conversione di un numero naturale in base `b`.

Ultima revisione: 17 novembre 2015.

Punti: 4.

Scrivete una funzione di prototipo

```
int inBase(int n, int b, char *s, int lun);
```

che converta il numero naturale `n` in base `b`, memorizzandone le cifre nella stringa `s`. Assumete che il parametro `lun` sia pari al numero delle cifre di `n` in base `b` più 1, e che `s` punti a un array di `lun+1` caratteri terminato da `'\0'`. La funzione restituisce `-1` senza compiere alcuna elaborazione nel caso in cui non valga `2 <= b <= 10`. Se invece questa condizione è soddisfatta, la funzione esegue la conversione richiesta e restituisce 0.

Scrivete una breve procedura `main` per testare la vostra implementazione.

Esercizio 2

Estrazione di un numero naturale da una stringa.

Punti: 10.

Scrivete una funzione di prototipo

```
int nextN(char *s, int *pos, int b)
```

che estragga dalla stringa di caratteri `s` il prossimo numero naturale ivi presente, espresso in base `b` a partire dalla posizione di indice `*pos` della stringa. Assumete che valga `2 <= b <= 10`. Al termine della chiamata, se l'elaborazione è andata a buon fine, la funzione restituisce il numero estratto, e aggiorna il valore del puntatore `pos` in modo che il suo valore sia l'indice del primo carattere nella stringa successivo all'ultima cifra del numero estratto. Più in dettaglio, il comportamento della funzione è il seguente.

- Se `s` è `NULL` la funzione restituisce `-1`.
- Altrimenti, la funzione scandisce la stringa `s` a partire dalla posizione `pos` fino a quando non incontra un carattere numerico. (Usate la funzione `int isdigit(char)` del file di intestazione `ctype.h`.) Se non ve n'è alcuno, la funzione restituisce `-1`.
- Assumendo ora che `i` sia l'indice del primo carattere numerico della stringa dopo `pos`, la funzione scandisce le posizioni della stringa da `i` in poi fino a quando i caratteri incontrati sono numerici, e memorizza le cifre corrispondenti come interi in un array locale ausiliario `int ar[]` di dimensione massima `MAXEL`. (Definite `MAXEL` come costante nel programma tramite una direttiva `#define`. Per esempio, ponete `MAXEL` pari a `64`.)
- Se, nell'esecuzione del punto precedente, la funzione incontra una cifra `d` che non soddisfa `d<b`, termina restituendo `-1`.
- La funzione usa l'array ausiliario `ar` per calcolare il valore del numero espresso in base[1] `b`. Poi restituisce il valore alla funzione chiamante, assicurandosi però che il valore contenuto nella cella di memoria puntata da `pos` sia stato aggiornato in modo da risultare l'indice della prima posizione in `s` immediatamente successiva all'ultima cifra del numero estratto.

[1] *Suggerimento.* La funzione `double pow(double x, double y)` del file di intestazione `math.h` computa x^y.

Si noti che la funzione deve poter essere chiamata più volte consecutivamente su una stessa stringa ed essere in grado di estrarre correttamente i numeri contenuti nella stringa, in successione. Per esempio, supponiamo che `s` sia

```
A,011ssfs0d111d
```

e che `*pos` valga 0. Allora le quattro chiamate consecutive

```
nextN(s, pos, 2);
nextN(s, pos, 2);
nextN(s, pos, 2);
nextN(s, pos, 2);
```

restituiscono, nell'ordine, `3,0,7,-1`. Si presti attenzione al fatto che il valore puntato da `pos` deve cambiare dopo ciascuna chiamata.

Scrivete una breve procedura `main` per testare la vostra implementazione.

Esercizio 3

Prima versione della funzione `main`.

Punti: 6.

Scrivete una procedura `main` che implementi il programma seguente.

Il programma riceve gli argomenti dalla riga di comando. Vi devono essere esattamente tre argomenti — oltre, come sempre, al nome del programma. Se questo non è il caso, il programma termina con un messaggio d'errore appropriato. Altrimenti, il programma interpreta il secondo e il terzo argomento sulla riga di comando come due interi b e b', convertendo gli argomenti appropriatamente. Se non vale $2 \leq b, b' \leq 10$, o se il programma non riesce a interpretare uno dei due parametri in questione come intero, il programma termina con un messaggio d'errore; altrimenti prosegue come descritto di seguito.

Il programma estrae dal primo argomento sulla riga di comando i numeri espressi in base `b` presenti nella stringa. A tal fine usa la funzione `nextN` dell'Esercizio 2, fino a quando essa non restituisca `-1`. Il programma visualizza sul terminale i numeri estratti, uno per riga, esprimendoli in base 10, e riportando accanto a ciascun numero estratto la corrispondente espressione in base b'. Fatto ciò, termina. Per convertire i valori restituiti da `nextN` in base b' il programma usa la funzione[2] `inBase` dell'Esercizio 1.

Si veda la Fig. 2 per qualche esempio d'esecuzione.

Esercizio 4

Seconda versione della funzione `main`.

[2] *Suggerimenti.* In `main`, fate attenzione ad allocare la memoria di volta in volta necessaria per la stringa `s` prima della chiamata a `inBase(n, b, s, lun)`. Per prima cosa calcolate `lun` prendendo il logaritmo appropriato di `n`. Aggiungete 1 per far spazio al terminatore `'\0'`, allocate la memoria con `void *malloc(size_t dim)` di `stdlib.h`, e inserite subito il terminatore al posto giusto. Nel file di intestazione `math.h` è definita la funzione `double log(double)` che calcola il logaritmo naturale. Ricordatevi di liberare la memoria allocata quando non serve più.

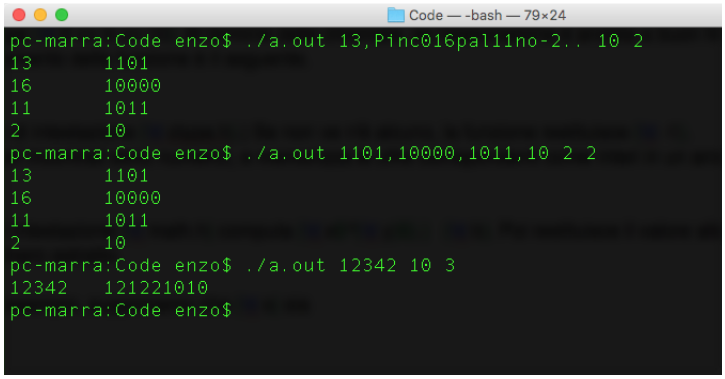

FIGURA 1. Esempio di esecuzione (Esercizio 3).

Punti: 8.

Estendete la procedura `main` dell'Esercizio 3 come segue. Se il numero degli argomenti sulla riga di comando non è tre o quattro, il programma termina con un messaggio d'errore.

Nel caso in cui vi siano tre argomenti sulla riga di comando, il programma si comporta esattamente come nell'Esercizio 3.

Se invece vi sono quattro argomenti sulla riga di comando, il programma interpreta i primi due come due basi $2 \leq b, b' \leq 10$, e gli ultimi due come due nomi di file, `filein` e `fileout`. Il programma apre il primo file, ed estrae tutti i numeri in base b contenuti nel file, scandendo il file riga per riga ed estraendo i numeri da ciascuna riga uno alla volta.[3] Il programma converte questi numeri in base b' e li scrive nel file `fileout`, uno per riga. Fatto ciò, il programma termina. Per quanto vi è possibile gestite gli errori nell'apertura e nella manipolazione dei file.

Si veda la Fig. 2 per qualche esempio d'esecuzione.

[3] *Suggerimento.* Usate la funzione `char *fgets(char *, int, FILE *)` del file di intestazione `stdio.h`.

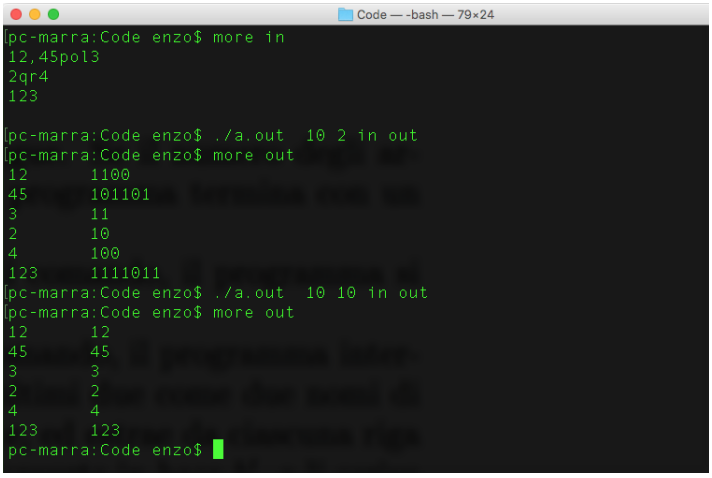

FIGURA 2. Esempio di esecuzione (Esercizio 4).

Istruzioni per la consegna

- Funzioni da consegnare (in un unico file sorgente).

  `main()`

  `inBase()`

  `nextN()`

  Tutte le eventuali altre funzioni ausiliarie che abbiate scritto.

- Sito per la consegna:

  `https://upload.mat.unimi.it`

- Autenticarsi con le proprie credenziali di posta elettronica d'ateneo.
- Consegnare solo il codice sorgente (file `*.c`), non l'eseguibile.

(V. Marra) DIPARTIMENTO DI MATEMATICA *Federigo Enriques*, UNIVERSITÀ DEGLI STUDI DI MILANO, VIA CESARE SALDINI, 50, I-20133 MILANO
E-mail address: `vincenzo.marra@unimi.it`