Prova d'esame di Laboratorio di Programmazione Strutturata
per il corso di laurea in Scienze e Tecnologia dei Media
Prova scritta – 5 Febbraio 2013

• **Esercizio 1**

Si scriva una *function* (la quale, eventualmente, richiama un'altra *function* anch'essa da scrivere) che effettua l'*ordinamento* di un vettore. Per fissare le idee, il prototipo della *function* richiesta è il seguente:

```
void ordina(int v[], int n);
```

dove per n s'intende il numero di elementi del vettore v , i quali (*al termine dell'esecuzione della function* `ordina`) dovranno essere disposti in ordine crescente.

All'interno di un commento della *function* `ordina`, si indichi chiaramente come varia la *complessità computazionale* dell'algoritmo prescelto in funzione di n .

• **Esercizio 2**

Si scriva una *function* il cui prototipo è il seguente

```
char *parola(char riga[], char *stringa);
```

che ha lo scopo di isolare la **prima parola** nel *vettore di caratteri* `riga` e lo riporta in `stringa`. Per fissare le idee, si consideri ad esempio che il contenuto di `riga` sia

		H	o		p	e	r	s	o			l	e		p	a	r	o	l	e	\n	\0

e si assumano le seguenti convenzioni:

1. al termine di una singola parola ci sia solo il carattere di *fine stringa* (cioè '\0') oppure carattere di "*a capo*" (cioè '\n') oppure *uno o più spazi*; per semplicità, non si considerano gli apostrofi, la punteggiatura, etc.;

2. si presti attenzione al fatto che all'inizio di `riga` possono trovarsi uno o più spazi;

3. all'interno della *function* `parola` devono essere **allocate** *le celle di memoria* del *vettore* `stringa`, le quali (alla fine dell'esecuzione della suddetta *function*) dovranno ospitare i *caratteri* che formano la **prima parola** individuata;

4. il *vettore di caratteri* `stringa` deve essere terminato da '\0';

5. la *function* `parola` deve restituire l'*indirizzo di memoria del primo carattere successivo alla fine della* **prima parola**;

6. qualora non ci fosse alcuna parola, allora la *function* `parola` deve restituire l'indirizzo di memoria (fittizio) `NULL`.

• **Esercizio 3 (facoltativo)**

Si scriva una *function* il cui prototipo è il seguente

```
void lunghezze(FILE *ifp, int nvolte[100]);
```

che ha lo scopo di **contare il numero di volte** che una parola di lunghezza l capita nel **file di testo** corrispondente all'indirizzo di memoria `ifp`. Per fissare le idee, si immagini che all'interno del **file di testo** corrispondente a `ifp` ci sia il testo di una nota canzone di Ligabue, che dopo (la riga riportata in precedenza) continua come segue:

o		f	o	r	s	e		s	o	n	o		l	o	r	o	\n	\0

	c	h	e		p	e	r	d	o	n	o		m	e	\n	\0

s	i		s	o	n	o		n	a	s	c	o	s	t	e		b	e	n	e	\n	\0

(a memoria non vado oltre la prima strofa, quindi terminiamo qui).
Si assumano le seguenti convenzioni:

1. si *deve utilizzare ripetutamente la function* `parola` descritta nel precedente esercizio 2; pertanto, si assumano anche tutte le regole già enunciate nel testo dell'esercizio 2;

2. si assuma che la lunghezza di ciascuna riga del **file di testo** corrispondente all'indirizzo di memoria `ifp` sia *inferiore a 100*;

3. la *function* `lunghezze` deve restituire il vettore `nvolte`, tale che (alla fine dell'esecuzione della *function*) l'elemento `nvolte[`l`]` deve essere uguale al numero di volte che è stata trovata una *parola di lunghezza* l , all'interno del **file di testo** corrispondente all'indirizzo di memoria `ifp`.