Fondamenti L-A 2006/2007 per Ing. Elettronica e delle Telecomunicazioni

Seconda prova parziale, Martedì 7 Dicembre 2006, ore 15, Aule 6.1–6.2

Nome:	**Cognome:**
Matricola:	**Compito:**

(scrivere i propri dati in stampatello)

Modalità di svolgimento della prova
(invariate rispetto alla prova in itinere)

- Scrivere nell'intestazione di questo foglio il proprio nome, cognome e numero di matricola.
- Utilizzare per le risposte i fogli protocollo allegati. Scrivere a penna. Scrivere su ciascun foglio protocollo il proprio nome, cognome e numero di matricola.
- Sui fogli protocollo, per ciascuna risposta indicare in modo chiaro il numero dell'esercizio corrispondente (es. *"esercizio 9"*).
- È assolutamente vietato consultare libri, appunti, manuali, o altro materiale didattico.
- Per tutta la durata dell'esame è necessario tenere spenti tutti i dispositivi elettronici e di comunicazione (cellulari, calcolatrici, palmari, smartphone, laptop, player, ...)
- Nell'ultima mezz'ora di prova non è consentito uscire dall'aula (nemmeno per andare in bagno).
- Per le parti non di codice, si accettano soluzioni scritte in italiano, inglese, portoghese, francese, spagnolo o turco.
- Al termine della prova, *consegnare assieme ai fogli protocollo anche il testo dell'esercizio.*

Tabella degli operatori in C

Precedenza	Operatori	Associatività
1	`() [] -> .`	a sinistra
2	`! ++ -- & *`	a destra
3	`* / %`	a sinistra
4	`+ -`	a sinistra
6	`< <= > >=`	a sinistra
7	`== !=`	a sinistra
11	`&&`	a sinistra
12	`||`	a sinistra
13	`?...:`	a destra
14	`= += -= *=`	a destra
15	`,`	a sinistra

Modalità di valutazione
(invariate rispetto alla prova in itinere)

Il compito si divide in tre parti (teoria, analisi, progetto+implementazione). *Non è necessario rispondere a tutte le domande*, ma per superare l'esame è necessario ottenere una valutazione superiore alla *soglia minima* in ciascuna parte. Il *massimo di punti ottenibili* in ciascuna parte è: 7 per le domande di teoria, 6 per la parte di analisi, 20 per la parte di progetto e implementazione. Il voto complessivo, se sufficiente, sarà compreso tra 15 e 33:

Parte A: Teoria [7 punti]
Soglia minima 4 punti

1. [punti $\in \{-1, +2\}$] **Record & puntatori**
2. [punti $\in \{-1, +2\}$] **Funzioni**
3. [punti $\in \{-1, +2\}$] **Run-time**
4. [punti $\in \{-1, +2\}$] **Binding**
5. [punti $\in \{-1, +1\}$] **File**

Parte B: Analisi [6 punti]
Soglia minima 3 punti

6. [punti $\in \{-1, +3\}$] **Output di funzione**
7. [punti $\in \{-1, +6\}$] **Evoluzione dello stack**

Parte C: Progetto & Implementazione [20 punti]
Soglia minima 9 punti

8. [punti $\in \{-1, +6\}$] **Progetto della soluzione**
9. [punti $\in \{-1, +1\}$] **Dichiarazioni**
10. [punti $\in \{-1, +2\}$] **Implementazione main**
11. [punti $\in \{-1, +5\}$] **Implementazione funzione #1**
12. [punti $\in \{-1, +8\}$] **Implementazione funzione #2**

Prototipi delle funzioni di uso comune

Header	Interfaccia	Error
string.h	`size_t strlen( char * ); // size_t e' un tipo int`	
string.h	`char *strcpy( char *, const char * ); // copia`	
string.h	`char *strcat( char *, const char * ); // concatenazione`	
string.h	`int strcmp( const char *, const char * ); // confronto`	
stdlib.h	`void *malloc( size_t ); // restituisce un generico puntatore`	`NULL`
stdlib.h	`void free( void * );`	
stdio.h	`FILE *fopen( char* name, char *mode );`	`NULL`
stdio.h	`int fclose( FILE * );`	`EOF`
stdio.h	`int feof( FILE * ); // vero se file pointer su EOF`	
stdio.h	`int fseek( FILE *, long offs, int orig ); // 0 SEEK_SET, 1 SEEK_CUR, 2 SEEK_END`	
stdio.h	`void rewind( FILE * );`	
stdio.h	`long ftell( FILE * ); // byte a cui si trova il file pointer`	`-1`
stdio.h	`int fprintf( FILE *, char * [, ...] );`	
stdio.h	`int fscanf( FILE *, char * [, ...] );`	
stdio.h	`char *fgets( char *, int, FILE * ); // legge la riga intera fino a '\n'`	`NULL`
stdio.h	`int *fputs( char *, FILE * ); // scrive una stringa e aggiunge '\n'`	`EOF`
stdio.h	`int fread( void *vet, int size, int n, FILE *fp );`	
stdio.h	`int fwrite( void *vet, int size, int n, FILE *fp );`	

Parte A: Teoria

1. Discutere l'uso degli operatori aritmetici (somma, sottrazione) con i puntatori in C, e in relazione ai vettori. Fare degli esempi.
2. Quali sono le caratteristiche di un servitore, nel modello cliente-servitore? Quando viene attivato? Quali parti rende visibili al cliente, e come?
3. Si spieghi cosa succede ogni volta che viene invocata una funzione, nel modello a *run-time* del C.
4. Legame per valore e legame per riferimento: si discutano queste due modalità di passaggio dei parametri, e si faccia un esempio di passaggio di un puntatore in C (il puntatore viene passato per valore o per riferimento?)
5. A che cosa serve il *file pointer*? Quando viene inizializzato? Come viene modificato dalle operazioni che si effettuano sui file?

Parte B: Analisi

6. Si scrivano l'output e il contenuto di `tempfile` a seguito dell'esecuzione del seguente codice:
```
FILE *fp; int i=0; char c[3];
int v[]={7,6,5,4,3,2,1};
fp=fopen( "tempfile", "w" );
while( v[i]-2 )
   fprintf( fp, "%d", v[i++] );
fclose( fp );
fp=fopen( "tempfile", "rb" );
while( fread( c, sizeof( char ), 2, fp ) )
   printf( "%c", c[0] );
fclose( fp );
```
7. Si descrivano l'evoluzione dello stack e l'output prodotto dall'esecuzione del seguente programma:
```
int f( int x, char c ) {
   if( x<5 ) return 1;
   if( x<10 ) return x;
   printf( "%c ", c+(x%5) );
   return f( x/2, c )+f( x-7, c );
}
int main() {
   printf( "%d", f( 22, 'a' ) );
}
```

Parte C: Progetto & Implementazione

Una compagnia aerea low-cost (*LC*) vi ha commissionato un programma per la gestione di voli internazionali. Per ciascun volo, *LC* gestisce svariate informazioni, tra cui:

- il *numero* del volo (rappresentato da una stringa di non più di 8 caratteri);
- la sigla dell'areoporto di partenza (stringa di 3 caratteri);
- la sigla dell'areoporto di arrivo (stringa di 3 caratteri);
- giorni della settimana in cui il volo viene effettuato.

Tutti i codici sono privi di spazi. Tali informazioni, da far leggere al programma, sono contenute al momento in un file di testo, chiamato `info_voli.txt`. `info_voli.txt` è strutturato per righe di testo: ogni riga contiene un numero di volo, i due codici degli areoporti, e infine i giorni della settimana in cui il volo viene effettuato. I giorni sono indicati con dei numeri progressivi da 1 a 7 (1=lunedì), mentre la lettera 'X' indica che il volo è temporaneamente sospeso.

Un esempio di file di testo è il seguente:
```
LC103 AMS BLQ 135
LC703 LHR AMS X
LC104 BLQ AMS 247
LC1002 LHR BLQ 1234567
```
Tale file indica che la compagnia vola da Amsterdam a Bologna di lunedì (`1`), mercoledì (`3`) e venerdì (`5`), da Bologna ad Amsterdam di martedì, giovedì e domenica (`247`), segnala la presenza di un volo (sospeso, `X`), codice LC703 da Heathrow ad Amsterdam, e di un volo invece attivo tutti i giorni della settimana (`1234567`), da Heathrow a Bologna. Si può notare come le righe non siano ordinate all'interno del file secondo alcun criterio. D'altro canto, si sa che non possono esistere due righe con la stessa partenza e arrivo. Infine, si può ipotizzare che le righe non saranno più di 500.

La *LC* avrà bisogno di poter interrogare il programma per ottenere le seguenti informazioni:

(a) elenco dei voli temporaneamente sospesi
(b) elenco di tutti i voli da/a un determinato areoporto (dato da input)
(c) elenco dei voli in cui sia l'andata sia il ritorno vengono effettuate in un determinato giorno della settimana (dato da input)

8. Si descriva in modo chiaro e succinto il progetto di una possibile soluzione software di tale problema: struttura della soluzione, funzioni e relativi input/output, variabili globali se usate, descrizione a parole o mediante diagramma di flusso dei punti salienti degli algoritmi che si intendono utilizzare.
9. Si forniscano le dichiarazioni in C di tutte le funzioni ed eventuali strutture dati globali necessarie all'implementazione della soluzione proposta.
10. Si implementi una funzione `menu` da usare per scegliere la funzionalità da far eseguire al programma
11. Si implementi una funzione da utilizzare in una possibile soluzione al problema proposto, in cui vengano lette dal file di input e predisposte nelle opportune strutture dati tutte le informazioni necessarie all'elaborazione dei dati stessi da parte del programma.
12. Si implementi in modo completo una funzione che risponda al punto (*b*) o al punto (*c*). *(Se la funzione si basa su dati organizzati su strutture dati interne al programma in un certo modo, è necessario anche fornire l'implementazione delle funzioni/procedure che effettuano tale organizzazione — a meno di quanto già mostrato al punto precedente.)*