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. Operatore di dereferencing e operatore indirizzo: spiegare cosa sono, come funzionano, e fare degli esempi.
2. Gli identificatori locali a una funzione, ovvero dichiarati nel corpo di essa, sono visibili all'esterno? È possibile usare tali identificatori per agire su dati visibili all'esterno? Fare uno o due esempi per chiarire.
3. Di cosa si compone un record di attivazione? Come viene gestito lo *stack* dei record di attivazione? In particolare: come avviene la procedura di allocazione/deallocazione di ciascun record su/dallo stack?
4. Si discutano le problematiche relative ai vettori come parametri di funzioni. Come avviene il "passaggio" di un vettore all'invocazione di una funzione? Si faccia un esempio.
5. Si consideri il problema della lettura di 10 record consecutivi da un file binario. Spiegare con un esempio quali sono le strutture dati necessarie per effettuare l'operazione di lettura all'interno di un programma in C.

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[]={77,103,32,10};
fp=fopen( "tempfile", "w" );
while( v[i]%8 )
   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 2;
   if( x<10 ) return x;
   printf( "%c ", c+(x%5) );
   return f( x/2, c )+f( x-8, c );
}
int main() {
   printf( "%d", f( 22, 'f' ) );
}
```

Parte C: Progetto & Implementazione

La Associazione Gestori di Osteria di Via del Pratello (*AGOViP*) vuole organizzare un *pub crawl* per festeggiare l'arrivo di Babbo Natale. Il *pub crawl* consiste in un percorso a tappe in cui ciascuna tappa è un'osteria. I partecipanti devono cercare di raggiungere la fine del percorso consumando una birra a ciascuna tappa.

Per ottenere una riuscita ottimale dell'iniziativa, l'*AGOViP* ha predisposto una serie di giornate di prova, in cui un gruppo di volontari equipaggiati con sensori, GPS, e apparecchi vari effettuano il *pub crawl* dall'inizio alla fine, mentre i loro dati vengono raccolti da una centralina e spediti a un computer. Nelle ultime settimane sono state effettuate 10 prove, in cui 20 volontari hanno effettuato un percorso abbreviato di 15 tappe. Ora l'*AGOViP* vi chiede di predisporre un programma software che sia in grado di visualizzare certe informazioni, a partire dai dati raccolti.

I dati sono salvati in un file binario, chiamato `pbcrwl.bir`. A mano a mano che arrivavano le informazioni dalla centralina (al superamento di ogni tappa del percorso da parte di un partecipante, ovvero a ciascuna uscita da osteria), il computer le archiviava aggiungendo un record in coda al file binario. I record hanno il seguente formato:

- un codice alfanumerico di 5 caratteri, che identifica univocamente il soggetto da cui sono estratte le informazioni;
- un intero $\in \{1, \ldots 10\}$ che identifica la prova effettuata;
- un intero $\in \{18, \ldots 95\}$ che rappresenta l'età del soggetto;
- un carattere 'P'/'M', che rappresenta la consumazione alla tappa (birra media o piccola)
- tre interi che rappresentano il tempo, in giorni, ore e minuti, trascorso dall'inizio della prova al momento della rilevazione.

L'*AGOViP* vi chiede un programma che sia in grado di leggere correttamente le informazioni delle varie prove, e che metta a disposizione le seguenti funzionalità:

(a) elenco (senza ripetizioni) dei partecipanti che hanno partecipato a più di una prova e che hanno più di una certa età (data da input);
(b) se ci sono ancora dei partecipanti in giro o dispersi (cioè se ci sono dei soggetti che hanno cominciato una prova ma non l'hanno completata;
(c) per una data prova, il cui identificatore è fornito in input, lista dei partecipanti che hanno impiegato meno del 20% del tempo impiegato dal più lento a terminare l'ultima tappa (se ce ne sono).

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.)*