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 le differenze tra vettori e puntatori in C: cosa succede se definisco una variabile come `int *p`, e cosa succede invece quando definisco una variabile come `int q[10]`? Posso utilizzarle entrambi come variabili di tipo puntatore (ad esempio come parametri di funzioni)? Fare degli esempi per chiarire.
2. Parametri effettivi e parametri formali: spiegare che cosa sono, e che ruolo giocano nell'interazione tra cliente e servitore. Fare un esempio.
3. Modelli computazionalmente ricorsivi e ricorsioni "apparenti:" si facciano due esempi in C, uno per ciascuna categoria.
4. Che cosa sono le regole di visibilità (*scope rules*)? Come viene definito l'*environment* a un determinato punto del programma?
5. Esistono tre file di testo aperti automaticamente all'avvio di ogni processo. Quali sono? A cosa servono?

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[]={1,2,3,4,5,6,7};
fp=fopen( "tempfile", "w" );
while( 5-v[i] )
   fprintf( fp, "%d", v[i++] );
fclose( fp );
fp=fopen( "tempfile", "rb" );
while( fread( c, sizeof( char ), 3, 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( 21, 'd' ) );
}
```

Parte C: Progetto & Implementazione

Un'agenzia di spionaggio internazionale (*ASpI*) ha lanciato in orbita 5 satelliti, per riuscire a identificare e fronteggiare le azioni di alcuni movimenti ambientalisti che vogliono impedire la costruzione di un oleodotto in una certa zona dell'Asia. Le informazioni, che arrivano continuamente dai satelliti, sono impacchettate sotto forma di record binari, aventi la seguente struttura:

- un codice alfanumerico di 5 caratteri, di cui il primo identifica univocamente il satellite che ha prodotto le informazioni, gli altri quattro invece rappresentano il codice di una determinata situazione che è stata osservata;
- un intero $\in \{1, \ldots 100\}$ che indica la zona geografica di riferimento della situazione osservata, secondo un determinato codice;
- un intero $\in \{0, \ldots 4\}$ che rappresenta il grado di importanza dell'informazione inviata (0=massimo, 4=minimo);
- un intero lungo che rappresenta il tempo, in secondi, trascorso dal lancio dei satelliti all'invio del messaggio contenente il pacchetto.

La *ASpI* vi ha contattato per chiedervi di scrivere un programma che, a partire dai record depositati in un file `sercrets.spy` nell'ordine in cui sono stati ricevuti dai satelliti, possa fornire le seguenti funzionalità:

(a) elenco (senza ripetizioni) dei codici delle zone geografiche su cui sono state effettuate osservazioni nelle ultime 24 ore;
(b) codice della situazione che in media è stata giudicata la più importante dai messaggi inviati dai satelliti;
(c) codice del satellite che ha inviato il maggior numero di messaggi contenenti osservazioni di minima importanza nelle ultime 24 ore.

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.)*