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

Quinto scritto, Martedì 26 Giugno 2007, ore 9, Aula 6.2

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

(scrivere i propri dati in stampatello)

Modalità di svolgimento della prova

- 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

Il compito si divide in quattro parti (due di teoria, una di analisi, una di progetto e implementazione). *Non è necessario rispondere a tutte le domande*, ma per superare l'esame è necessario ottenere una valutazione sufficiente (cioè superiore alla *soglia minima*) in ciascuna parte.

Il *massimo di punti ottenibili* in ciascuna parte è: 4 per ciascuna sezione di teoria, 6 per la parte di analisi, 19 per la parte di progetto e implementazione. Il voto complessivo, se sufficiente, sarà compreso tra 18 e 33.

Parte A: Teoria #1 [4 punti]
Soglia minima 2 punti

1. [punti $\in \{-1, +4\}$] **Architettura**
2. [punti $\in \{-1, +4\}$] **Linguaggi**

Parte B: Teoria #2 [4 punti]
Soglia minima 2 punti

3. [punti $\in \{-1, +4\}$] **Variabili**
4. [punti $\in \{-1, +4\}$] **Funzioni**

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

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

Parte D: Progetto & Implementazione [19 punti]
Soglia minima 9 punti

7. [punti $\in \{-1, +7\}$] **Progetto della soluzione**
8. [punti $\in \{-1, +1\}$] **Prototipi e dichiarazioni di tipo**
9. [punti $\in \{-1, +2\}$] **Implementazione** `menu`
10. [punti $\in \{-1, +6\}$] **Implementazione funzione #1**
11. [punti $\in \{-1, +7\}$] **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

Massimo una pagina

1. Si presenti la gerarchia delle memorie e si evidenzino le caratteristiche che giustificano la scelta di un tipo di memoria/tecnologia piuttosto che un altro per le varie funzionalità che si intendono realizzare.
2. Si scriva una grammatica BNF da utilizzare per la produzione di stringhe nel formato: `N323.`, `M132.`, `M1234.`, `N3214.`, `N1333.`, etc. (`N` o `M` seguita da tre o quattro cifre nell'intervallo [1..4], seguite da `.`).

Parte B: Teoria #2

Massimo una pagina

3. Si mostri come viene ripartita la memoria assegnata all'esecuzione di un programma C (codice/dati) e si spieghi quali porzioni della memoria vengono allocate alle varie categorie di dati (variabili globali, locali alle funzioni, dinamiche, parametri delle funzioni, variabili `static`).
4. Si spieghi cosa avviene a *run-time* quando viene invocata una funzione. Si descriva il contenuto del record di attivazione. In particolare, si consideri una funzione il cui prototipo prevede un solo parametro p, che rappresenta l'*indirizzo* di una variabile, e che definisca al suo interno una variabile (locale) x di tipo intero. Si discutano gli eventuali "effetti collaterali" della chiamata a funzione.

Parte C: Analisi

5. Si scriva l'output prodotto dall'esecuzione del seguente programma:

```
int f( int j, char c ) {
   printf( "%c", c );
   return j-2;
}

int main() {
   int i=6;
   while( i>0 ) {
      printf( "%d\n", i=f( i, 'B' + i )
);
   }
}
```

6. Si descrivano l'evoluzione dello stack e l'output prodotto dall'esecuzione del seguente programma:

```
int start[]={2,3,4,5,6};

int fun( int *v, int i )
{
   int j=i-1, sum=v[0];
   if( i>0 ) {
      while( j>0 ) {
         sum *= v[j--];
      }
      printf( "%d\n", fun( v+1, i-1)
);
   }
   return sum;
}

int main() {
   printf( "%d\n", fun( start, 4)
);
}
```

Parte D: Progetto & Implementazione

Un'Agenzia di Promozione Turistica (APT) vuole mettere a disposizione del pubblico un programma per raccogliere e fornire informazioni su percorsi di trekking. I percorsi sono suddivisi in tre categorie di difficoltà (facile/medio/impegnativo), e hanno una durata di percorrenza stimata in ore. Ciascun percorso ha un nome, che lo distingue univocamente dagli altri percorsi. L'APT vi chiede di sviluppare tale programma. Il programma deve fare uso di un file per conservare i dati inseriti in modo permanente. Inoltre, il programma deve fornire le seguenti funzionalità:

(a) **inserimento** di un nuovo percorso nel database. Questa funzionalità deve verificare che non sia stato già definito un percorso con lo stesso nome; nel caso, deve chiedere di specificare un nuovo nome per il percorso che si vuole inserire;

(b) **visualizzazione** dell'elenco dei percorsi che soddisfano certi criteri di difficoltà e durata. L'utente deve essere in grado di specificare una *difficoltà massima* e/o una *durata massima*, e il programma deve visualizzare nome, difficoltà e durata di *tutti* i percorsi che soddisfano i criteri.

(c) **cancellazione** di un percorso, a partire dal nome.

In fase di avvio il programma deve essere in grado di recuperare lo stato salvato in precedenza.

7. Si imposti la soluzione. Si fornisca una descrizione accurata degli algoritmi e delle strutture dati necessari per implementare il programma.
8. Si forniscano i prototipi in C delle funzioni ed eventuali dichiarazioni di tipo necessarie all'implementazione della soluzione proposta.
9. Si implementi in C una funzione `menu` da usare per selezionare la funzionalità da far eseguire al programma e per effettuare l'eventuale salvataggio su file dei dati aggiornati, ove l'utente scelga di terminare l'esecuzione del programma.
10. Si scriva il codice in C di una funzione parametrica che implementi la funzionalità (a) del programma.
11. Si scriva il codice in C di una funzione parametrica che implementi la funzionalità (b) del programma.