Politecnico di Milano - Anno Accademico 2003-2004 - Informatica C Seconda prova in itinere – 5 Luglio 2004 – **Elaborato B**	
COGNOME e NOME	
Matricola	

DOMANDA SU TEMI DI LABORATORIO

Si supponga che il seguente programma venga scritto esattamente come sotto riportato e che si cerchi poi di compilarlo e di eseguirlo. Stabilire quanti e quali errori contiene e indicarli, dopodichè, per ogni errore trovato:

- indicare come lo si potrebbe correggere.
- Specificare se si tratta di un errore che il compilatore rileverebbe non appena si tentasse di compilare il programma, oppure di un errore che si manifesterebbe successivamente, in fase di esecuzione.

```
void main()
{

  int x:                          : invece di ;  (compile time)

  controlla_numero(x);            la funzione viene chiamata prima
                                  che sia stata definita o dichiarata
}                                 (compile time)

void controlla_numero(int x)
{
                                  X non è mai inizializzato (RUN)
  if(x=1)printf("il numero vale uno\n");     = invece di == !
                           (warning a compile time, errore a run time)
  else printf("il numero non vale uno\n");

}
```

Politecnico di Milano - Anno Accademico 2003-2004 - Informatica C Seconda prova in itinere – 5 Luglio 2004 – **Elaborato B**	
COGNOME e NOME	
Matricola	

Domande su argomenti della II parte del corso

1. Files (punti 5)

E' stato richiesto di scrivere un programma che copia il contenuto di un file su un altro file, con le seguenti caratteristiche:

- **Non viene copiato l'intero contenuto del primo file, ma solo un carattere sì e uno no**; in altre parole, del primo file vengono copiati sul secondo file solo i caratteri di posto dispari: il primo carattere del file, il terzo carattere, il quinto,...; vengono invece trascurati i caratteri di posto pari. Per esempio, se il file origine conteneva solo la parola "Pippo", il file destinazione conterrà "Ppo". In generale quindi il file destinazione risulta lungo circa la metà di quello origine.
- **Al termine della copiatura vengono chiusi i file origine e destinazione.**

Si assuma che la parte iniziale del programma, quella che verifica i parametri e che apre i file origine (**`infile`**) e destinazione (**`outfile`**), sia già stata scritta.

Completare il programma sotto riportato in modo tale che si comporti esattamente come sopra descritto. E' ammesso solo fare delle aggiunte e solo nei punti del programma ove compaiono gli appositi riquadri; le parti fuori dai riquadri non possono assolutamente essere modificate.

```
#include <stdio.h>
int main( ... )
{
  FILE * infile;
  FILE * outfile;
  int count=1;
  char c;

  /* .. la parte iniziale del programma, comprendente le operazioni di verifica
parametri e apertura file, è già stata scritta .. */
```

```
while(!feof(infile))
{
  c=fgetc(infile);
  if(count % 2 == 1)
      fputc(c, outfile);
  count ++;
}
 OPPURE
while(!feof(infile))
{
  fscanf(infile, "%c", &c);
  if(count % 2 == 1)
      fprintf(outfile, "%c", c);
  count ++;
}
```

Ciclo copiatura – punti 4

```
fclose(infile);
fclose(outfile);
```

Chiusura files – punti 1

```
}
```

Politecnico di Milano - Anno Accademico 2003-2004 - Informatica C Seconda prova in itinere – 5 Luglio 2004 – **Elaborato B**	
COGNOME e NOME	
Matricola	

2. Liste (punti 5)

Un archivio di dati personali utilizza come struttura dati una lista doppia che contiene attualmente 4 nodi. Ogni nodo contiene nome, cognome, indirizzo ed età di una persona. Il puntatore alla testa della lista è la variabile globale **`head`**, il puntatore alla coda della lista è la variabile globale **`tail`**.

I nodi sono di un tipo così definito:

```
struct nodo_s {
      char nome[30];
      char cognome[30];
      char indirizzo[50];
      int eta;
      struct nodo_s * next;
      struct nodo_s * prev;
};
```

Percorrendola dalla testa alla coda, la lista contiene i dati delle quattro persone i cui cognomi sono Bianchi, Gialli, Neri e Rossi (in quest'ordine).

Data una variabile x di tipo int e una variabile y di tipo char,

- **scrivere un'istruzione che assegna a x il valore dell'età di Rossi. Non usare il puntatore tail. (scandire la lista dalla testa) –** ***punti 1***

```
x=head->next->next->next->eta;
```

- **scrivere un'istruzione che assegna a x la lunghezza del nome di Gialli. Non usare il puntatore head. (scandire la lista dalla coda) –** ***punti 1***

```
x=strlen(tail->prev->prev->nome);
```

- **scrivere un'istruzione che assegna a y il secondo carattere dell'indirizzo di Neri. (si assuma che tale indirizzo sia sicuramente lungo almeno 2 caratteri) –** ***punti 1***

```
y=head->next->next->indirizzo[1];
```

- **supponendo che sia già stato creato (allocato e compilato) un nodo per il sig. Zefiro, raggiungibile mediante il puntatore `nuovonodo`, scrivere le istruzioni necessarie per inserire tale nodo in ultima posizione nella lista. –** ***punti 2***

```
tail->next=nuovonodo;
nuovonodo->prev=tail;
tail=nuovonodo;
nuovonodo->next=NULL;
```

Politecnico di Milano - Anno Accademico 2003-2004 - Informatica C Seconda prova in itinere – 5 Luglio 2004 – **Elaborato B**	
COGNOME e NOME	
Matricola	

3. Algoritmi (punti 3)

Scrivere una funzione **`int mystrlen(char * stringa)`** che prende una stringa come parametro e ritorna un valore pari alla lunghezza della stringa ricevuta senza usare alcuna funzione di libreria. Implementare la stessa funzione in due modi diversi, utilizzando prima un algoritmo iterativo (punti 1) e poi uno ricorsivo (punti 2).

Suggerimento per l'implementazione ricorsiva: *data una stringa di N (N>0) caratteri, la sua lunghezza è in generale pari a [1+ [lunghezza della stringa composta dai suoi ultimi N-1 caratteri]]…*

Implementazione iterativa

```
int mystrlen_iterativa(char * stringa)
{
    int i;
    for(i=0; stringa[i]!='\0'; i++);

    return i;
}
```

Implementazione ricorsiva

```
int mystrlen_ricorsiva(char * stringa)
{
  if(stringa[0]=='\0')return 0;
  else return 1+mystrlen_ricorsiva(stringa+1);
}
```

4. Puntatori (punti 2)

La variabile **`buffer`** è stata dichiarata come un array di caratteri (destinato a contenere una stringa). Scrivere un ciclo for in cui ogni carattere della stringa viene trasformato in maiuscolo (sfruttando ogni volta la funzione toupper); scrivere il ciclo SENZA FARE USO dell'operatore [] per l'accesso all'array e SENZA FARE USO della funzione strlen.

```
for(i=0; *(buffer+i)!=0; i++)
    *(buffer+i) = toupper(*(buffer+i));
```

5. Teoria (punti 2)

Spiegare come può accadere che un'area di memoria utilizzata in precedenza dal programma risulti non più raggiungibile.

L'unico modo è per le aree allocate dinamicamente (malloc). Se il puntatore con cui si può raggiungere l'area dati perde il suo valore (con un assegnamento, oppure essendo dichiarato in una funzione che termina, o per altre ragioni …) allora l'area di memoria NON E' PIU' RAGGIUNGIBILE, (non avendo alcun nome!)