Politecnico di Milano - Anno Accademico 2005-06 - Informatica C
Appello 13 settembre 2006 – A

COGNOME e NOME	
Matricola	

Es 1. [punti 5]: Files

Scrivere una funzione che apre in lettura il file input.txt e in scrittura il file output.txt (creandolo se non esiste già e sovrascrivendolo se esiste già), dopodichè legge il file input.txt, che contiene un elenco di numeri interi, uno per riga, e per ciascun numero letto calcola il doppio e lo stampa su output.txt andando a capo ogni volta. Al termine devono essere chiusi entrambi i files. Se invece non era stato possibile aprire uno dei due files, il programma deve segnalarlo e terminare.

```
#include <stdio.h>

int main(int argc, char *argv[])
{
  FILE * infile;
  FILE * outfile;
  int num;
  infile=fopen("input.txt","r");
  if(infile==NULL){printf("no input file");return;}
  outfile=fopen("output.txt","w");
  if(outfile==NULL){printf("no output file");return;}
  while(!feof(infile))
  {
     if(fscanf(infile,"%d",&num)<0)break;
     fprintf(outfile,"%d\n",num*2);
  }
  fclose(infile); fclose(outfile);
  system("pause");
  return 0;
}
```

Politecnico di Milano - Anno Accademico 2005-06 - Informatica C
Appello 13 settembre 2006 – A

COGNOME e NOME	
Matricola	

Es 2. [punti 3]

Scrivere il main di un programma prova.exe che, se è stato specificato un numero dispari di parametri su riga di comando, stampa la lunghezza del primo di essi (inteso come stringa) e termina. Se invece il numero di parametri ricevuti è pari il programma non stampa nulla. Il comportamento deve essere come quello qui accanto mostrato.

```
C:\>prova ciao pippo

C:\>prova ciao
4

C:\>prova

C:\>prova ciao pippo pluto
4
```

```
int main(int argc, char *argv[])
{
  int i,cnt=0;
  if(argc%2==0)
    printf("%d\n",strlen(argv[1]));
  return 0;
}
```

Politecnico di Milano - Anno Accademico 2005-06 - Informatica C
Appello 13 settembre 2006 – A

COGNOME e NOME	
Matricola	

Es 3a. [punti 3]

Definire un tipo di dato idoneo a fungere da nodo di lista singola con capacità di contenere alcuni dati anagrafici di una persona, consistenti in via (stringa), città (stringa), telefono (stringa), CAP (stringa) ed età (numero). Per le stringhe evitare lunghezze eccessive ove è possibile prevedere con esattezza la massima lunghezza necessaria. Per i numeri usare il tipo di dato piu' appropriato per evitare di sprecare memoria e per rappresentare valori del tipo adatto.
Definire inoltre una variabile **`head`**, di tipo idoneo a contenere un puntatore alla testa della lista.

```
struct nodo_s{
        char via[30];
       char citta[30];
       char telefono[15];
       char CAP[6];
       unsigned char eta; /* piccoli interi positivi (<255) */
        struct nodo_s * next;
};

struct nodo_s * head;
```

Es 3b. [punti 4]

Scrivere una funzione elimina, che, assumendo che la lista sia gia' popolata con almeno un elemento, elimina l'elemento in testa, lasciando la lista rimanente in uno stato coerente e accessibile.

```
int elimina()
{
      struct nodo_s * scan;
      scan=head->next;
        free(head);
        head=scan;
      return 0;
}
```

Politecnico di Milano - Anno Accademico 2005-06 - Informatica C
Appello 13 settembre 2006 – A

COGNOME e NOME	
Matricola	

Es 4. [totale max punti 3]

Ogni risposta esatta vale 1 punto. Ogni risposta errata vale -0.25 punti.
Ogni risposta non data vale 0 punti.
NOTA: Nel caso di quesiti a risposta chiusa, tra le risposte proposte per la domanda una sola e' esatta.

4a

```
int a;
a=72516;
printf("%d\n",a);
```

[ ] il frammento di programma e' corretto
[x] il frammento di programma e' corretto solo su macchine in cui gli interi sono a 32 bit o piu'
[ ] il frammento di programma e' rifiutato dal compilatore

4b

```
int a,b;
a=10;
b=a/3;
printf("%d\n",b);
```

[ ] il frammento di programma contiene un errore sintattico
[ ] il frammento di programma è sintatticamente corretto ma in esecuzione si verifichera' un errore che potra' provocarne l'uscita
[x] il frammento di programma è sintatticamente corretto ma in esecuzione si verifichera' una perdita di precisione nei calcoli a causa di un troncamento

4c

Descrivere (e motivare) il comportamento di questo frammento di programma.

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    int i;
    for(i=0; i!=3; i+=2)
    {
       printf("%d\n",i);
    }
  return 0;
}
```

Il programma non termina, in quanto il test per l'uscita dal ciclo non scatta mai: l'indice di ciclo assume solo valori pari e non potra' mai valere 3. Vengono stampati i numeri pari da 0 in su.