Politecnico di Milano - Anno Accademico 2005-06 - Informatica C
Appello 10 luglio 2006 – B

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 e ne trascrive su output.txt *i soli caratteri di posto dispari*, ossia il primo, il terzo, il quinto e così via. I caratteri di posto pari vengono ignorati. Per esempio, se input.txt contiene la frase "prova testo da convertire", al termine dell'esecuzione output.txt dovrà contenere "poatsod ovrie". 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 car;
  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,"%c",&car)<0)break;
     fprintf(outfile,"%c",car);
     if(fscanf(infile,"%c",&car)<0)break;
  }
  fclose(infile); fclose(outfile);
  system("pause");
  return 0;
}
```

Politecnico di Milano - Anno Accademico 2005-06 - Informatica C
Appello 10 luglio 2006 – B

COGNOME e NOME	
Matricola	

Es 2. [punti 3]

Scrivere il main di un programma prova.exe che, se è stata specificata esattamente una stringa su riga di comando, ne stampa la lunghezza e il numero di "a" contenute (contando insieme minuscole e maiuscole), altrimenti segnala che il numero di parametri è sbagliato e termina immediatamente. Il comportamento deve essere come quello indicato qui accanto.

```
C:\>prova.exe prova
Lunghezza=5
Numero di 'a' contenute=1

C:\>prova.exe niente
Lunghezza=6
Numero di 'a' contenute=0

C:\>prova.exe
Numero errato di parametri
```

```
int main(int argc, char *argv[])
{
  int i,cnt=0;
  if(argc!=2)
  {
    printf("Numero errato di parametri\n");
    return;
  }
  printf("Lunghezza=%d\n",strlen(argv[1]));
  for(i=0; i<strlen(argv[1]); i++)
    if(tolower(argv[1][i])=='a')cnt++;
  printf("Numero di 'a' contenute=%d\n",cnt);
  return 0;
}
```

Politecnico di Milano - Anno Accademico 2005-06 - Informatica C
Appello 10 luglio 2006 – B

COGNOME e NOME	
Matricola	

Es 3a. [punti 2]

Definire un tipo di dato idoneo a fungere da nodo di lista singola con capacità di contenere valori di tipo intero.
Definire inoltre una variabile **`head`**, di tipo idoneo a contenere un puntatore alla testa della lista.

```
struct nodo_s{
        int valore;
        struct nodo_s * next;
};

struct nodo_s * head;
```

Es 3b. [punti 5]

scrivere una funzione **`inserisci_in_testa`** che prende un parametro di tipo intero, non ritorna parametri, può accedere alla variabile globale head sopra dichiarata, e provvede a costruire un nodo caricato con il valore specificato e a inserire tale nodo in testa alla lista.

```
void inserisci_in_testa(int valore)
{
      struct nodo_s * tmp;
      tmp=(struct nodo_s *) malloc(sizeof(struct nodo_s));
      tmp->valore=valore;
      tmp->next=head;
      head=tmp;
}
```

Politecnico di Milano - Anno Accademico 2005-06 - Informatica C
Appello 10 luglio 2006 – B

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

```
    char buffer[100];
    printf("Immettere una parola ");
    scanf("%s",&buffer);
    printf("La parola immessa e' %s\n",buffer);
```

[ ] il frammento di programma sarà rifiutato dal compilatore
[x] il frammento di programma sarà accettato dal compilatore ma in esecuzione potrà causare errori
[ ] il frammento di programma e' corretto

4b

```
    char buffer[100];
    strcpy(buffer,"prova");
    printf("%d",strlen(buffer)<<2);
```

Il frammento di programma produce il seguente risultato: 20

4c

```
#include <stdio.h>
void elabora(int i)
{
     i*=2;
     return i;
}

int main(int argc, char *argv[])
{
    int i=30000;
    printf("%d",elabora(i));
}
```

[x] il frammento di programma sarà rifiutato dal compilatore
[ ] il frammento di programma sarà accettato dal compilatore ma in esecuzione potrà causare errori
[ ] il frammento di programma e' corretto