Politecnico di Milano - Anno Accademico 2001-2002 - Informatica B Recupero – 25 Febbraio 2002 – **Elaborato B**	
NOME COGNOME	
Matricola	

REC2 /Punti	I / 2 punti	II / 2 punti	III / 2 punti	IV /3 punti	V /3 punti	REC1
Tot= / 12						

REC2 I – Calcolatore – Completare un semplice programma in C e in Assembler che conta quanti dei 3 numeri già contenuti in una tabella TAB sono **diversi da 0** e ne scrive il numero. Usare le frasi già predisposte (per semplificare; si consiglia di **NON stravolgere** l'impostazione) (2 punti)

Assembler			*Linguaggio C*
EXIT	EQU	2000	Void main()
NEQ	DW	? /* non equal	int NEQ;
PORTA_Video	EQU	2	
VOLTE	DW	? /* cont ciclo	int VOLTE;
TAB	DW	2	int TAB[3]={2,0,8};
	DW	5	
	DW	0	
START	LDA	#0	{ VOLTE = 0 ;
	STA	VOLTE	
	STA	NEQ	NEQ = 0 ;
CICLO	LDA	VOLTE	while (VOLTE != 3)
	SUB	#3	
	JZ	FINITO	{
	LDA	N	If (N != TAB[VOLTE])
	LDI	VOLTE	
	SUB	TAB(I)	
	JZ	UGUALI	
	LDA	NEQ	NEQ=NEQ+1;
	ADD	#1	
	STA	NEQ	
UGUALI	LDA	VOLTE	VOLTE=VOLTE+1;
	ADD	#1	
	STA	VOLTE	
	JMP	CICLO	} * end while
FINITO	LDA	NEQ	Scrivi NEQ ;
	OUT	PORTA_Video	
	JMP	EXIT	} * end main

Politecnico di Milano - Anno Accademico 2001-2002 - Informatica B Recupero – 25 Febbraio 2002 – **Elaborato B**	
NOME COGNOME	
Matricola	

REC2 - II – cicli e funzioni (2 punti)

Si supponga già dichiarato l'array quadrato `int tabella[][]`.
Scrivere una funzione `void riempi_array_tabella(int n)` che genera la tavola del prodotto di ordine `n`, riempiendo le celle dell'array `tabella` con gli opportuni valori, supponendolo dichiarato come array quadrato di `n` per `n` numeri interi. Per esempio, se l'array fosse 3x3, il suo contenuto dopo l'esecuzione della funzione dovrebbe essere il seguente:

1	2	3
2	4	6
3	6	9

```
#include <stdio.h>
int tabella[10][10];
void riempi_array_tabella(int n)
{
     int i,j;
     for(i=0; i<n; i++)
          for(j=0;j<n;j++)
               tabella[i][j]=(i+1)*(j+1);
}
void main()
{
     riempi_array_tabella(10);
}
```

REC2 - III – Files (2 punti)

Il **file di testo** `inp.txt` contiene righe di testo (in quantità non nota a priori) del seguente tipo:

ciao 46 101.25
prova 15 99.41

ciascuna di tali righe contiene tre campi: una stringa (una parola priva di spazi e lunga al massimo 8 caratteri), un numero intero e un numero con parte frazionaria. I tre campi sono separati da spazi.

Il programma da completare, sotto riportato, deve leggere da inp.txt tali informazioni e generare in output il **file binario** `exit.dat` nel quale le stesse informazioni vengono scritte sotto forma di strutture conformi alla seguente definizione:

```
typedef struct {char s[9]; int a; double b;}EL;
```

Per esempio partendo da un file inp.txt contenente le due righe di esempio sopra riportate il programma dovrebbe generare in uscita un file exit.dat contenente due strutture riempite come segue:

NOME COGNOME	
Matricola	

s	a	b
ciao	46	101.25
prova	15	99.41

Potrà essere utile l'uso della funzione `fscanf()`.

```
main()
{FILE *fin, *fout;
 EL elem;

 /* apertura files – completare! */
 fin=fopen("inp.txt","r");
 fout=fopen("exit.dat","wb");

 if((fin!=NULL) && (fout!=NULL))
   { /* ciclo lettura-scrittura fino a esaurimento del file fin - completare! */
     fscanf(fin, "%s%d%lf", elem.s, &(elem.a), &(elem.b));
      while(!feof(fin))
           { fwrite(&elem,sizeof(elem),1,fout);
             fscanf(fin, "%s%d%lf", elem.s, &(elem.a), &(elem.b));}
      fclose(fin); fclose(fout);
     }
 else
   {  /* segnalazione errore – completare! */
      printf("errore apertura file!\n");
   }
 return(0);
}
```

REC2 - IV – liste (3 punti)

Data una lista, che potete supporre contenga **esattamente 3 elementi**, basata sulla seguente definizione:

```
struct nodo_s { int dato; struct nodo_s * next; }
struct nodo_s * head;
```

scrivere un frammento di codice che assegna al valore contenuto nel secondo nodo della lista lo stesso valore contenuto nel primo nodo della lista solo se il valore contenuto nel terzo nodo è maggiore di 30

```
if(head->next->next->dato->  > 30)
     head->next->dato=head->dato;
```

scrivere le istruzioni necessarie per eliminare l'ultimo (terzo) nodo della lista

```
struct nodo_s *tmp=head->next->next;
head->next->next=NULL;
free(tmp);
```

Politecnico di Milano - Anno Accademico 2001-2002 - Informatica B
Recupero – 25 Febbraio 2002 – Elaborato B

NOME COGNOME	
Matricola	

REC2 - V – Programma (3 punti)

Un modo per approssimare i valori della funzione esponenziale è dato dalla seguente serie:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots \text{ ossia: } e^x = 1 + x + \frac{x^2}{1\times 2} + \frac{x^3}{1\times 2\times 3} + \frac{x^4}{1\times 2\times 3\times 4} + \cdots$$

il valore approssimato così calcolato è tanto più preciso quanti più termini della serie si sommano; ci si può quindi limitare a prendere in considerazione tanti termini quanti bastano per ottenere la precisione desiderata.
Per esempio, ci si può fermare quando si riscontra che la differenza fra il valore ottenuto sommando K termini e il valore ottenuto sommando K+1 termini non supera, in valore assoluto, un valore desiderato (cioè la precisione richiesta).

Scrivere il programma che, facendo riferimento alla formula sopra indicata ed alle istruzioni fornite, calcola il valore di e^x per un x inserito da tastiera, approssimando il risultato alla seconda cifra decimale (ovvero in modo che la differenza fra due valori consecutivi della serie troncata sia minore in valore assoluto di 0,001).

Può essere utile usare la funzione `double fabs(double val)` che dato un valore ne restituisce il valore assoluto e, ma non è indispensabile, `double pow(double base, double esponente)` che restituisce il valore base^{esponente}. Saranno opportunamente valutate le soluzioni più efficienti.

```
double ex(double x)
  { int passo=1;
    double prec=0.0, att=1.0, pot=1.0;

    while(fabs(att-prec)>0.001)
      { prec=att;
     pot = pot *x/passo;
     att = att + pot;
     passo=passo+1;
      }
    return(2*prec);
  }
```

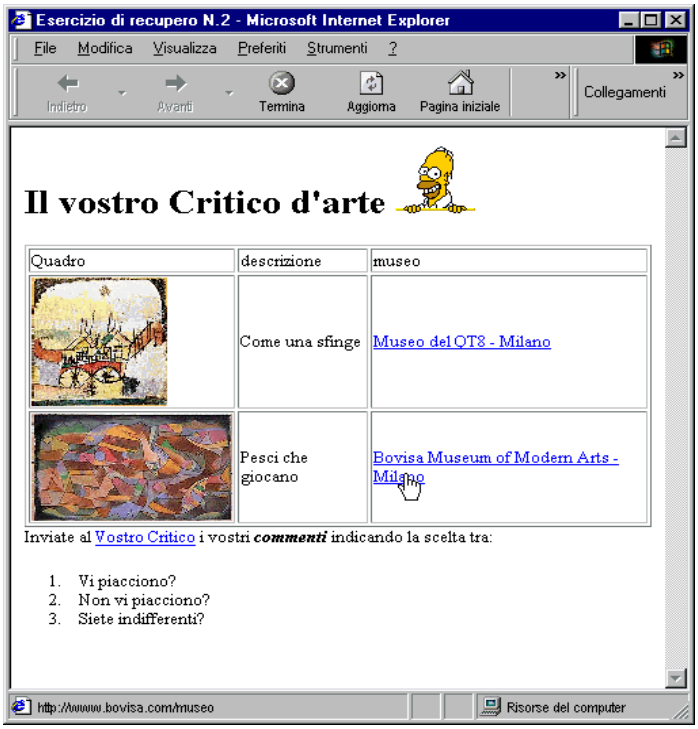
Politecnico di Milano - Anno Accademico 2001-2002 - Informatica B
Recupero – 25 Febbraio 2002 – Elaborato B

NOME COGNOME	
Matricola	

REC1 – Recupero Primo Compitino

Programmare una pagina HTML che appaia come in figura. (usare le stesse frasi e raggiungere lo stesso effetto)

1. il titolo della pagina sia **Esercizio di Recupero N.2**
2. la pagina ha una intestazione di massima grandezza seguita dall'immagine **homer.gif**
3. una tabella centrata ha **una riga di intestazione** e due righe di contenuti
4. in ogni riga **le tre colonne contengono**
   a. l'immagine del quadro **"titolo.gif"**
   b. il titolo del quadro
   c. un link **al museo in cui è custodito il quadro del tipo www.nomemuseo.com** (ogni museo ha una pagina che si chiama **"museo.html"**
5. sotto la tabella appare un invito a scrivere al Critico i commenti (la parola Critico è il link per la posta elettronica homer@accademia.com la parola "***commenti***" à in italico e grassetto
6. IL tipo di commento che si deve inviare è **una lista numerata** con tre possibilità di scelta



NOME COGNOME	
Matricola	

```
<html>
<head>
     <title>Esercizio di recupero N.2</title>
</head>
<body>
<h1> Il vostro Critico d'arte <img SRC="homer.gif" ></h1>
<center><table  >
<tr>
<th>Quadro</th>
<th>descrizione</th>
<th>museo</th>    </tr>

<tr>
<td><img SRC="come_una_sfinge.jpg" ></td>
<td>Come una sfinge</td>
<td><a href="http://www.qt8.com/museo.html">Museo del QT8 -
Milano</a></td>     </tr>

<tr>
<td><img SRC="pesci_che_giocano.jpg" ></td>
<td>Pesci che giocano</td>
<td><a href="http://www.bovisa.com/museo.html">Bovisa Museum of Modern
Arts - Milano </a></td>     </tr>
</table></center>

Inviate al <a href="mailto:homer@accademia.com">Vostro Critico</a> i
vostri <i><b>commenti</b> </i>indicando la scelta tra:

<ol>
<li>  Vi piacciono?</li>
<li>  Non vi piacciono?</li>
<li>  Siete indifferenti?</li>
</ol>

</body>
</html>
```