	Politecnico di Milano Scuola di Ingegneria Industriale e dell'Informazione **FONDAMENTI DI INFORMATICA** Appello 16 Settembre 2016	COGNOME E NOME
		MATRICOLA
		Spazio riservato ai docenti

- Il presente plico contiene 4 esercizi e **deve essere debitamente compilato con cognome e nome, numero di matricola**.
- Il tempo a disposizione è di 1 ora e 40 minuti.
- Non separate questi fogli. Scrivete la soluzione solo sui fogli distribuiti, utilizzando il retro delle pagine in caso di necessità. Cancellate le parti di brutta con un tratto di penna.
- Ogni parte non cancellata a penna sarà considerata parte integrante della soluzione.
- È possibile scrivere a matita (e non occorre ricalcare al momento della consegna!).
- **È vietato utilizzare telefoni o pc**. Chi tenti di farlo vedrà annullata la sua prova.
- È ammessa la consultazione di libri e appunti.
- **Qualsiasi tentativo di comunicare con altri studenti comporta l'espulsione dall'aula**.
- È possibile ritirarsi senza penalità.
- **Non è possibile lasciare l'aula conservando il tema della prova in corso**.

Esercizio 1 (10 punti)

Il **checksum**, è un semplice meccanismo per controllare la correttezza di un messaggio inviato attraverso una rete informatica che consiste nel calcolare la somma dei bit che lo compongono (che corrisponde al numero di 1 contenuti nella codifica) ed inviarla insieme al messaggio.

A. Si definisca un tipo di dato, **messaggio**, in grado di contenere una stringa ed il suo valore di checksum.
B. Si implementi la funzione, **verifica,** che riceve in ingresso una variabile di tipo **messaggio** e restituisce 1 se il checksum è corretto e 0 viceversa.

Nota. Il checksum di una stringa deve essere calcolato sommando i bit della codifica binaria di ciascun carattere che la compongono. Si suggerisce di implementare una funzione di supporto, **chksum**, che riceve in ingresso un char e restituisce il checksum della sua codifica binaria.

Esempio. La stringa "hi" è composta dal carattere 'h' e il carattere 'i'. La codifica ASCII di 'h' è 104 (1101000 in binario), mentre quella di 'i' è 105 (1101001 in binario). Il checksum di 'h' è quindi 3, mentre quello di 104 è 4. Il checksum del messaggio è quindi 3+4=7.

Soluzione

A.

```
typedef struct
{
     char msg[1000];
     int checksum;
} messaggio;
```

B.

```
int chksum(char c)
{
     int sum = 0;
     while (c>0)
     {
          sum += c%2;
          c /= 2;
     }

     return sum;
}

int verifica(messaggio m)
{
     int sum=0;
     for (int i=0; i<strlen(m.msg); i++)
          sum += chksum(m.msg[i]);

     return sum==m.checksum;

}
```

Esercizio 2 (10 punti)

Un array di interi si definisce **positivo** se ciascuno dei suoi elementi è maggiore o uguale di 0, **strettamente positivo** se ciascuno dei suoi elementi è maggiore di 0. Ad esempio, {4,3,3,2,1} è strettamente positivo, mentre {4,5,0,2,1} è positivo, infine {4,5,-3,2,1} non è ne positivo ne strettamente positivo.

Implementare in C una funzione **ricorsiva** per determinare se un array è positivo o strettamente positivo. La funzione riceve in ingresso un array di interi e la sua dimensione; ritorna 1 se l'array è strettamente positivo, 0 se l'array è positivo, -1 altrimenti.

Nota. Un array vuoto è strettamente positivo.

Soluzione

```
int positivo (int a[], int n)
{
     int tmp;

     if (n<=0)
          return 1;

     tmp = positivo(a+1,n-1);
     if (tmp == -1 || a[0]<0)
          return -1;
     if ( (a[0]==0 && tmp>=0) || (a[0]>0 && tmp==0) )
          return 0;
     if (a[0]>0 && tmp==1)
          return 1;

}
```

Esercizio 3 (8 punti)
Utilizzando il linguaggio assembly introdotto a lezione, implementare un programma che legge dal nastro di ingresso una sequenza di numeri interi finché il numero inserito è maggiore del precedente; non appena tale condizione non è più verificata, il programma scrive sul nastro di uscita il numero di interi letti (ad esclusione dell'ultimo) e termina la sua esecuzione.

Specifica in C del programma da implementare (utilizzare il nastro di ingresso e il nastro di uscita per le operazioni di lettura e scrittura su stdin e stdout):

```
scanf("%d",&x);
scanf("%d",&y);
n=1;

while (y>x)
{
      n++;
      x=y;
      scanf("%d",&y);
}

printf("%d",n);
```

Soluzione

```
LOAD= 1
STORE 101
READ
STORE 102
READ
STORE 103
SUB 102
BLE 15
LOAD 101
ADD= 1
STORE 101
LOAD 103
STORE 102
BR 5
LOAD 101
WRITE
END
```

Esercizio 4 (5 punti)
Si consideri la seguente funzione in C

```
int f (int v[], int n, int e)
{
    int *p;
    for (p=v; p-v<n; p++)
        if (*p == e)
            return p-v;
    return -1;
}
```

Riportare che cosa ritorna la chiamata `f(v,n,e)` nei seguenti casi (giustificando brevemente la risposta):

A. v={1,3,4,2,9}; n=5; e=2;
B. v={1,3,4,2,9}; n=2; e=2;
C. v={1,3,4,2,9}; n=5; e=5;

Soluzione
La funzione riceve un vettore di interi **v**, la sua dimensione **n**, ed un elemento **e**. Quindi cerca l'elemento **e** nel vettore **v** e, se lo trova, restituisce l'indice corrispondente alla posizione all'interno del vettore in cui si trova; altrimenti, restituisce il valore -1.

A. La funzione ritorna 3, che è l'indice del valore 2 all'interno del vettore
B. La funzione ritorna -1, poichè non torva il valore 2 all'interno dei primi due elementi di **v** (dato che la dimensione del vettore che viene passata è 2).
C. La funzione ritorna -1, poichè **v** non contiene il valore 5