	Politecnico di Milano, Facoltà di Ingegneria Industriale **INFORMATICA B** Prova in itinere del 20 Novembre 2013		COGNOME E NOME
	RIGA	COLONNA	MATRICOLA
			Spazio riservato ai docenti

- Il presente plico contiene 3 esercizi e deve essere debitamente compilato con cognome e nome, numero di matricola, posizione durante lo scritto (comunicata dal docente).
- Il tempo a disposizione è di 1 ora e 15 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** (o ripudiate) 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!) **facendo in modo comunque che quanto scritto sia ben leggibile**.
- È **vietato** utilizzare **calcolatrici, telefoni o pc**. Chi tenti di farlo vedrà **annullata** la sua prova.
- È ammessa la consultazione di **libri** e **appunti**, purché con pacata discrezione e senza disturbare.
- **Non è permesso consultare temi d'esame degli anni precedenti**
- 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 1A (6 punti)

Sia data la definizione di stringa,

```
typedef char Stringa[30];
```

la seguente struttura di utente Twitter:

```
typedef struct {
    Stringa nome;
    int     data_iscrizione; /* anno di iscrizione a twitter */
    int     numero_messaggi; /* numero effettivo messaggi scritti dall'utente*/
    Tweet   messaggi[100];
} Utente;
```

e la definizione di un messaggio tweet di 140 caratteri, contenente al massimo 4 utenti menzionati:

```
typedef struct {
    char contenuto[140];
    Stringa menzionati[4]; /* contiene i nomi di utenti twitter preceduti da @*/
    int  numero_menzionati; /* numero di utenti menzionati effettivamente nel messaggio*/
} Tweet;
```

Si consideri infine, il seguente database di utenti:

```
typedef struct {
    Utente dati_utente[100]; /* contiene i dati di al massimo 100 utenti*/
    int numero_utenti; /* numero effettivo di utenti*/
}Database;
```

e la dichiarazione della seguente variabile

```
Database twitterdb;
```

Si chiede di scrivere un frammento di codice C per stampare a video tutti gli utenti iscritti dal 2008 (escluso) che hanno menzionato '@obama' in almeno uno dei propri messaggi. Se, ad esempio, vi sono solo due utenti nel database:

@vzaccaria, iscritto dal 2012, messaggi:

"@obama for president!"

"@obama elected!"

@merkel, iscritta dal 2012, messaggi:

"Come to germany!"

Il programma deve stampare:

Utente @vzaccaria ha menzionato @obama !!

NB. Per controllare gli utenti menzionati in un Tweet non è necessario scorrere il contenuto perché questi sono salvati nel campo menzionati

Soluzione

```
int main()
{
    int u;
    int t;
    int h;
    for(u=0; u<twitterdb.numero_utenti; u++)
    {
        int trovato = 0;
        if(twitterdb.dati_utente[u].data_iscrizione > 2008)
        {
            for(t=0; t < twitterdb.dati_utente[u].numero_messaggi; t++ )
            {
                for( h=0; h< twitterdb.dati_utente[u].messaggi[t].numero_menzionati &
!trovato; h++)
                {

if(strcmp(twitterdb.dati_utente[u].messaggi[t].menzionati[h],"@obama")==0)
                        trovato = 1;
                }
            }
        }
        if(trovato)
        {
            printf("Utente %s ha menzionato @obama !!\n", twitterdb.dati_utente[u].nome);
        }
    }
}
```

Esercizio 1B (6 punti)

Sia data la definizione di stringa:

```
typedef char Stringa[30];
```

e la definizione di un messaggio tweet di 140 caratteri, contenente al massimo 4 hashtags (parole in un tweet che cominciano con '#', ad esempio, "#poli"):

```
typedef struct {
    char contenuto[140];
    Stringa hashtags[4]; /* contiene solamente gli hashtags presenti nel messaggio*/
    int  numero_hashtags; /* numero effettivo di hasthtags presenti nel messaggio*/
} Tweet;
```

Sia data la seguente struttura di utente twitter:

```
typedef struct {
    Stringa nome;
    int     data_iscrizione; /* anno di iscrizione a twitter */
    int     numero_messaggi; /* numero effettivo di messaggi scritti dall'utente*/
    Tweet   messaggi[100];
} Utente;
```

e la seguente definizione del database di utenti:

```
typedef struct {
   Utente dati_utente[100]; /* contiene i dati di al massimo 100 utenti*/
    int numero_utenti; /* numero effettivo di utenti*/
} Database;
```

Si consideri la dichiarazione della seguente variabile

```
Database twitterdb;
```

e si scriva un frammento di codice C per stampare a video tutti gli utenti iscritti dal 2011 (escluso) che hanno usato l'hashtag "#poli" in almeno uno dei propri messaggi. Se, ad esempio, vi sono solo tre utenti nel database:

@vzaccaria, iscritto dal 2012, messaggi:

"E' finito il primo emisemestre! #poli"

"E' ora del primo compitino! #poli"

@obama, iscritto dal 2009, messaggi:

"Politecnico is cool! #poli"

@merkel, iscritta dal 2012, messaggi:

"Come to germany!"

Il programma deve stampare:

Utente @vzaccaria ha usato #poli !!

NB. Per controllare gli hashtag presenti in un Tweet non è necessario scorrere il contenuto perché questi sono salvati nel campo hashtags

```
int main()
{
    int u;
    int t;
    int h;
    for(u=0; u<twitterdb.numero_utenti; u++)
    {
        int trovato = 0;
        if(twitterdb.dati_utente[u].data_iscrizione > 2011)
        {
            for(t=0; t< twitterdb.dati_utente[u].numero_messaggi; t++ )
            {
                for(h=0; h< twitterdb.dati_utente[u].messaggi[t].numero_hashtags; h++)
                {
                if(strcmp(twitterdb.dati_utente[u].messaggi[t].hashtags[h], "#poli")==0)
                       trovato = 1;
                }
            }
        }
        if(trovato)
        {
            printf("Utente %s ha usato #poli !!\n", twitterdb.dati_utente[u].nome);
        }
    }
}
```

Esercizio 2A (7 punti)

Si scriva un programma in linguaggio C per individuare le posizioni dei massimi locali all'interno di in un vettore **v** (contenente numeri razionali). Un elemento dell'array è un massimo locale se è strettamente maggiore dell'elemento precedente e seguente. Le posizioni dei massimi locali devono essere salvate in un secondo vettore **pos.**

In particolare il programma dovrà:

1. Richiedere all'utente quanti numeri intende inserire (massimo 100).
2. Popolare di conseguenza il vettore **v**,
3. Salvare in **pos**, senza lasciare buchi, e quindi stampare a schermo le posizioni dei massimi locali. Le posizioni dei massimi locali sono date dagli indici dell'array corrispondenti ai massimi locali.

N.B. il primo e l'ultimo elemento dell'array sono massimi locali se strettamente maggiori del valore adiacente.

Esempio,

si assuma che il vettore **v** contenga i seguenti elementi

v = **10**, 2, **4**, 3.5, **5.2**, 5, **7.34**, 3, 12, 12, 4, 1, **23.2**

il vettore **pos** contiene, nelle prime posizioni, 0, 2, 4, 6, 12

Soluzione

```
#include <stdio.h>
#define N 30

void main()
{
float v[N];
int pos[N];
int i, j, n;
float temp;


do
{
    printf("\ninserire n: ");
    scanf("%d", &n);
}while(n<0 || n > N);


for(i = 0; i < n; i++)
 {
    printf("\n v[%d] = ", i);
    scanf("%f", &v[i]);
 }


j = 0;
if(v[0] > v[1])
 {
   pos[j] = 0;
            j++;
 }

for(i = 1; i < n - 1; i++)
{
    if(v[i] > v[i - 1] && v[i] > v[i + 1])
        {
            pos[j] = i;
            j++;
        }
}

if(v[n-1] > v[n-2])
 {
   pos[j] = n-1;
   j++;
 }

printf("\n");
for(i = 0; i < j; i++)
    printf(" %d ", pos[i]);

}
```

Esercizio 2B (7 punti)

Si scriva un programma in linguaggio C per individuare le posizioni di segmenti costanti all'interno di in un vettore **v** (contenente numeri razionali). Un elemento appartiene ad un segmento costante se è uguale all'elemento precedente o all'elemento seguente. Le posizioni dei segmenti costanti devono essere salvate in un secondo vettore **pos.**

In particolare il programma dovrà:

1. Richiedere all'utente quanti numeri intende inserire (massimo 100),
2. Popolare di conseguenza il vettore **v**,
3. Salvare in **pos**, senza lasciare buchi, e quindi stampare a schermo, le posizioni dei segmenti costanti. Le posizioni dei segmenti costanti sono date da tutti gli indici degli elementi appartenenti a segmenti costanti.

N.B. il primo e l'ultimo elemento dell'array appartengono ad aree costanti se coincidono al valore adiacente.

Esempio,

si assuma che il vettore **v** contenga i seguenti elementi

v = **1**, **1**, 3.5, **5.1**, **5.1**, 7.34, 3, **12.2**, **12.2**, 4.2, 1, 23.2

il vettore **pos** contiene, nelle prime posizioni, 0,1,3,4,7,8

Soluzione

```
#include <stdio.h>
#define N 100

void main()
{
float v[N];
int pos[N];
int i, j, n;
float temp;

do
{
    printf("\ninserire n: ");
    scanf("%d", &n);
}while(n<0 || n > N);


for(i = 0; i < n; i++)
 {
    printf("\n v[%d] = ", i);
    scanf("%f", &v[i]);
 }


j = 0;

if(v[0] == v[1])
{
    pos[j] = 0;
    j++;
}

for(i = 1; i < n -1 ; i++)
{
    if(v[i] == v[i - 1] || v[i] == v[i + 1])
        {
            pos[j] = i;
            j++;
        }
}

if(v[n - 2] == v[n - 1])
{
    pos[j] = n -1;
    j++;
}

printf("\n le aree costanti appaiono nella posizione: ");
for(i = 0; i < j; i++)
    printf(" %d ", pos[i]);
}
```

Esercizio 3A (4 punti)

Si definisca il minimo numero di bit necessari per rappresentare in complemento a 2 tutti i seguenti valori

149, 108, 12, 42, 92

e si motivi adeguatamente la risposta.

Si eseguano quindi le seguenti operazioni riportando eventuali bit di carry (tra parentesi tonde) e di overflow (tra parentesi quadre).

i) - 149 – 108
ii) 108 + 12

Si ricorda che il bit di overflow è pari ad 1 se si verifica overflow, 0 altrimenti.

Soluzione

Sono necessari 9 bit per rappresentare tutti i numeri: 9 bit coprono l'intervallo $[-2^8, 2^8-1] = [-256, 255]$, mentre con 8 bit copriremmo l'intervallo $[-2^7, 2^7-1] = [-128, 127]$ e questo non basterebbe per il 149.

149 in binario è 10010101
-149 in CP2 a 9 bit diventa quindi 101101011

108 in binario è 1101100
108 in CP2 a 9 bit diventa quindi 001101100
-108 in CP2 a 9 bit diventa quindi 110010100

12 in binario è 1100
12 in CP2 a 9 bit è quindi 000001100

```
-149     101101011
-108     110010100
        -----------
   [1](1)011111111
```

C'è overflow perché la somma di due numeri negativi ha dato un numero positivo

```
  12     000001100
 108     001101100
        ----------
  [0](0) 001111000
```

Non c'è overflow, la somma di due numeri positivi ha dato un numero positivo.

Esercizio 3B (4 punti)

Si consideri la rappresentazione in virgola fissa con parte intera in CP2. Si specifichi il minimo numero di bit per la parte intera e per la parte razionale per i seguenti numeri:

i) 2
ii) 0.25
iii) -2
iv) -2.25

e si ricavi quindi la codifica equivalente.

Fissata tale codifica, si rappresenti il numero: 0.2 e si stabilisca se tale rappresentazione è esatta e, nel caso non lo fosse, si specifichi quanti bit occorrerebbero per esprimerlo accuratamente.

Soluzione

Per rappresentare i numeri sopra in CP2 servono m=3 bit per la parte intera e n =2 bit per la parte razionale.

Con 3 bit si in CP2 si copre l'intervallo $[-2^2, 2^2-1] = [-4, 3]$, mentre con 3 bit si copre l'intervallo [-2, 2 -1] = [2,1] che non basta per rappresentare il 2.

Per la parte razionale bastano 2 bit in quanto 0.25 (i.e., 1/4) viene codificato in 0.01.

La rappresentazione di questi numeri con m =3, n = *2* diventa

i) 2 -> 010.00
ii) 0.25 -> 000.01
iii) -2 -> 110.00
iv) -2.25 -> 110.01

Non è possibile rappresentare 0.2 con un numero finito di cifre in base 2 perché questo da luogo ad una rappresentazione periodica con periodo 0011.