	Politecnico di Milano Facoltà di Ingegneria Industriale **INFORMATICA B** Prova in itinere del 25 Novembre 2015		COGNOME E NOME
	RIGA	COLONNA	MATRICOLA

Tema A

Spazio riservato ai docenti

- Il presente plico contiene **3 esercizi** e deve essere debitamente compilato con cognome e nome, numero di matricola e posizione durante lo scritto (comunicata dal docente).
- Il tempo a disposizione è di 2 ore.
- 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) assicurandosi 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 tuttavia possibile 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 1 (7 punti)

Si consideri la seguente definizione di tipi di dato per rappresentare alcune nazioni, visitatori e padiglioni di EXPO2015:

```
#define MAX_CARAT 50 /* Dimensione del tipo Stringa */
#define MAX_PREF 3 /* Numero di padiglioni preferiti */
#define MAX_VIS 500 /* Numero massimo di visitatori in coda a un padiglione */
#define MAX_PAD 6 /* Numero di padiglioni */

typedef enum{italia, giappone, nordcorea, emiratiarabi, francia, cile} Nazione;

typedef char Stringa[MAX_CARAT];

typedef struct {
    Stringa nome, cognome;
    Nazione preferiti[MAX_PREF];
    double prezzo;
} Visitatore;

typedef struct {
    Stringa nome;
    Nazione ident;
    int ore_visita, minuti_visita;
    int len_coda;
    Visitatore coda[MAX_VIS]; // array dei visitatori attualmente in coda
} Padiglione;
```

Ogni visitatore è caratterizzato da un nome, un cognome, tre padiglioni preferiti e un prezzo pagato per il biglietto di ingresso a EXPO2015. Ogni padiglione è caratterizzato da un nome, dalla nazione di appartenenza, da un tempo di visita (in ore e minuti), dal numero di visitatori in coda (ovvero dalla lunghezza della coda) e dall'elenco di tutti i visitatori in coda.

A. 1) Si dichiari una variabile *EXPO* per contenere al massimo 6 padiglioni e si inizializzi il primo come padiglione con nome "Italia", caratterizzato da un tempo di visita di 3h e 5min.
   2) Si dichiari una variabile *primo* per contenere i dati del primo visitatore e si scriva una porzione di codice per richiedere all'utente i dati di primo.
   3) Si inserisca quindi primo nella coda del padiglione "Italia", aggiornando opportunamente i campi del padiglione.

Si assuma che la variabile *EXPO* sia stata popolata con *n_pad* padiglioni (numero intero, positivo e minore di *MAX_PAD*) e che i padiglioni contengano anche la lista dei visitatori in coda. Si scrivano porzioni di codice per risolvere le seguenti richieste, ricordandosi di dichiarare tutte le variabili che si ritiene necessario utilizzare.

B. 1) Si scriva una porzione di codice per individuare tutti i visitatori che sono in coda a un padiglione tra i loro preferiti.
   2) Inserire i dati di questi visitatori in un vettore visitatori_felici da dichiarare opportunamente.
   3) Si calcoli quindi il prezzo totale pagato dai visitatori così selezionati.

C. Si modifichi la dichiarazione del tipo Padiglione per associarvi anche il ristorante del

padiglione. In particolare, si definisca un tipo Ristorante per contenere le seguenti informazioni:

- nome
- costo del coperto
- prezzo medio di un pasto
- numero di coperti totale
- lista dei visitatori che vi stanno mangiando (massimo 50)
- numero di posti attualmente occupati

Soluzione

A) 1)

```
Padiglione EXPO[MAX_PAD];
int n_pad = 0;
int i;

strcpy(EXPO[0].nome,”Italia”);
EXPO[0].ident = italia;
EXPO[0].ore_visita = 3;
EXPO[0].minuti_visita = 5;
n_pad++;
```

2)

```
Visitatore primo;

printf("Inserire il nome del visitatore: \n");
scanf("%s",primo.nome);
fflush(stdin);
printf("Inserire il cognome del visitatore: \n");
scanf("%s",primo.cognome);
fflush(stdin);
for (i = 0; i < MAX_PREF; i++) {
        printf("Inserire il %d padiglione preferito: \n",i+1);
        printf("0: Italia, 1 Giappone, 2 Corea del Nord, 3 Emirati Arabi, 4 Francia, 5 Cile \n");
        scanf("%d",&primo.preferiti[i]);
}
printf("Inserire il prezzo del biglietto: \n");
scanf("%f",&primo.prezzo);
```

3)

```
EXPO[0].len_coda = 0;
EXPO[0].coda[EXPO[0].len_coda] = primo;
EXPO[0].len_coda++;
```

B)

```
Visitatore visitatori_felici[MAX_PAD * MAX_VIS];
int n_felici = 0;
double incasso = 0;

int i,j,h, no_stop;

for (i = 0; i < n_pad; i++)
```

```
        for (j = 0; j < EXPO[i].len_coda; j++) {
                /* i padiglioni preferiti di ogni visitatore sono tutti diversi. */
                h = 0;
                no_stop = 1;
                while (no_stop && h < MAX_PREF) {
                        if (EXPO[i].coda[j].preferiti[h] == EXPO[i].ident) {
                                visitatori_felici[n_felici] = EXPO[i].coda[j];
                                incasso += visitatori_felici[n_felici].prezzo;
                                n_felici++;
                                no_stop = 0;
                        }
                        h++;
                }
        }
```

C)

```
#define MAX_COPERTI 50

typedef struct{
        Stringa nome;
        float costo_coperto;
        float prezzo_medio;
        int n_coperti;
        Visitatori occupanti[MAX_COPERTI];
        int n_occupati;
} Ristorante;

typedef struct {
    Stringa nome;
    Nazione ident;
    int ore_visita, minuti_visita;
    Visitatore coda[MAX_VIS];
    int len_coda;
    Ristorante rist;
} Padiglione;
```

Esercizio 2 (6 punti)

Scrivendone prima lo pseudo codice, si sviluppi un programma in linguaggio C che svolga le seguenti operazioni:

1. **Definisca un tipo matrice** di interi di dimensione *MAX_R* e *MAX_C*, con *MAX_R* e *MAX_C* definite come costanti
2. **Dichiari una variabile *m*** del tipo matrice definito e tutte le altre variabili necessarie ai fini dell'esercizio
3. **Acquisisca da tastiera i valori** da assegnare alle celle della matrice *m*
4. **Chieda all'utente di fornire il numero della colonna** su cui effettuare l'operazione al punto seguente
5. **Calcoli la media** degli elementi contenuti nella matrice, nella **colonna** indicata dall'utente
6. **Copi, per righe, all'interno di un vettore** (monodimensionale) di dimensioni adeguate **tutti i dati contenuti nella matrice *m*** (in questa fase il programma dovrà inserire nel vettore prima i valori contenuti nella prima riga della matrice, poi di seguito quelli contenuti nella seconda riga, e così via).
7. **Esegua la stessa operazione indicata al punto 5** (calcolo della media dei valori nella colonna indicata della matrice) **lavorando** però **sul vettore** invece che sulla matrice.
8. **Controlli** che il valore calcolato al punto 7 e quello calcolato al punto 5 siano uguali. **In caso affermativo**, stampi "*ok, il valore della media e*`" seguito dal valore calcolato. **In caso negativo**, stampi "*errore!*" seguito dai due valori calcolati.

Soluzione

1) Pseudo codice

```
i ← 0
finché (i < MAX_R) esegui
        j ← 0
        finché (j < MAX_C) esegui
                leggi(m[i][j])
                j ← j+1
        chiudi esegui
        i ← i + 1
chiudi esegui

colonna  ← -1;
finché (colonna < 0 || colonna >= MAX_C) esegui
        leggi(colonna)
chiudi finché

media1 ← 0
i ← 0
finché (i < MAX_R) esegui
        media1 ← media1 + m[i][colonna]
chiudi finché


k ← 0
```

```
i ← 0
finché (i < MAX_R) esegui
        j ← 0
        finché (j < MAX_C) esegui
                vett[k] ← m[i][j]
                j ← j+1
                k ← k+1
        chiudi esegui
        i ← i + 1
chiudi esegui

media2 ← 0
k ← colonna
finché (k < MAX_R * MAX_C) esegui
        media2 ← media2 + vett[i]
        k ← k + MAX_C
chiudi esegui
media2 ← media2 / MAX_R

se (media1 = media2) allora
        stampa(“ok, il valore calcolato è “)
        stampa(media1)
altrimenti
        stampa(“errore!”)
        stampa(media1)
        stampa(media2)
chiudi se
```

2) Codice C

```
#include <stdio.h>

/* Si fa l’ipotesi che la matrice sia di dimensioni 3x5 */
#define MAX_R 3
#define MAX_C 5

typedef int matrice[MAX_R][MAX_C];
typedef int vettore[MAX_R * MAX_C];

int main()
{
  matrice m;
  vettore v;
  int i, j, k, colonna;
  float media1, media2;

  printf("Inserisci i valori della matrice %dx%d: ", MAX_R, MAX_C);
  for(i = 0;  i < MAX_R; i++)
    for(j = 0; j < MAX_C; j++)
      scanf("%d", &m[i][j]);
```

```
  printf("Inserisci il numero della colonna: ");
  scanf("%d", &colonna);
  while((colonna >= MAX_C) || (colonna < 0))
    scanf("%d", &colonna);

  media1 = 0;
  for(i = 0; i < MAX_R; i++)
    media1 = media1 + m[i][colonna];
  media1 = media1 / MAX_R;

  k = 0;
  for(i = 0; i < MAX_R; i++)
    for(j = 0; j < MAX_C; j++)
    {
      v[k] = m[i][j];
      k = k + 1;
    }

  media2 = 0;
  for(k = colonna; k < MAX_R * MAX_C; k = k + MAX_C)
    media2 = media2 + v[k];
  media2 = media2 / MAX_R;

  if(media1 == media2)
    printf("ok, il valore calcolato e` %f\n", media1);
  else printf("errore! %f %f\n", media1, media2);
  return(0);
}
```

Esercizio 3 (4 punti)
Si codifichi in IEEE (754-1975) a 32 bit i seguenti numeri

- 1.2
- 1.75
- 2.95

Indicando quali di questi ha una rappresentazione esatta.

Domanda extra:
Si consideri il seguente frammento di programma. La sua esecuzione porta a stampare
"La somma di 1.2 e 1.75 non è uguale a 2.95"
Si giustifichi brevemente come mai avviene questo.

```c
#include <stdio.h>

int main() {
  float dato1, dato2;

  dato1 = 1.2;
  dato2 = 1.75;

  printf("\n\nLa somma di %f + %f ", dato1, dato2);
  if (dato1+dato2 != 2.95) printf("non ");
  printf("e' uguale a %f\n\n", 2.95);

  return 0;
}
```

Soluzione:

1.2 in binario con codifica IEEE 754-1975 a 32 bit è pari a:

- Segno: 0
- Mantissa: 1.00110011001100110011001 e non necessita di essere normalizzata ed è periodica
- Esponente: 127 -> 01111111

1.75 in binario con codifica IEEE 754-1975 a 32 bit è pari a:

- Segno: 0
- Mantissa: 1.11000000000000000000000 e non necessita di essere normalizzata
- Esponente: 127 -> 01111111

2.95 in binario con codifica IEEE 754-1975 a 32 bit è pari a:

- Segno: 0
- Mantissa: 10.11110011001100110011001 = 1.01111001100110011001100 x 2^1
- Esponente: 127 + 1 = 128 -> 10000000

Per sommare i due numeri dovremo semplicemente sommare le mantisse, dal momento che l'esponennte è lo stesso. Avremo come mantissa: 1.11110011001100110011001, che non è uguale alla mantissa della rappresentazione di 2.95. Il tutto è dovuto quindi a due arrotondamenti differenti operati su 1.2 e 2.95.

	Politecnico di Milano Facoltà di Ingegneria Industriale **INFORMATICA B** Prova in itinere del 25 Novembre 2015		COGNOME E NOME
	RIGA	COLONNA	MATRICOLA
Tema B			Spazio riservato ai docenti

- Il presente plico contiene **3 esercizi** e deve essere debitamente compilato con cognome e nome, numero di matricola e posizione durante lo scritto (comunicata dal docente).
- Il tempo a disposizione è di 2 ore.
- 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) assicurandosi 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 tuttavia possibile 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 1 (7 punti)

Si consideri la seguente definizione di tipi di dato per rappresentare alcune nazioni, visitatori e padiglioni di EXPO2015:

```
#define MAX_CARAT 50  /* Dimensione del tipo Stringa */
#define MAX_PREF 3 /* Numero di padiglioni preferiti */
#define MAX_VIS 300 /* Numero massimo di visitatori in coda a un padiglione */
#define MAX_PAD 5  /* numero Numero di padiglioni */

typedef enum{italia, giappone, nordcorea, emiratiarabi, cile} Nazione;

typedef char Stringa[MAX_CARAT];

typedef struct {
    Stringa nome, cognome;
    Nazione preferiti[MAX_PREF];
    double prezzo;
} Visitatore;

typedef struct {
    Stringa nome;
    Nazione ident;
    int len_coda;
    int ore_visita, minuti_visita;
    Visitatore coda[MAX_VIS]; // array dei visitatori attualmente in coda
} Padiglione;
```

Ogni visitatore è caratterizzato da un nome, un cognome, tre padiglioni preferiti e un prezzo pagato per il biglietto di ingresso a EXPO2015. Ogni padiglione è caratterizzato da un nome, dalla nazione di appartenenza, da un tempo di visita (in ore e minuti), dal numero di visitatori in coda (ovvero dalla lunghezza della coda) e dall'elenco di tutti i visitatori in coda.

A. 1) Si dichiari una variabile *EXPO* per contenere al massimo 5 padiglioni e si inizializzi il primo come padiglione con nome "Giappone", caratterizzato da un tempo di visita di 4h e 25min.
   2) Si dichiari una variabile *primo* per contenere i dati del primo visitatore e la si inizializzi con i vostri dati.
   3) Si inserisca quindi tale visitatore nella coda del padiglione del Giappone, aggiornando opportunamente i campi del padiglione. (N.B. i campi di *primo* non devono essere inseriti dall'utente, ma è richiesto di scrivere opportune istruzioni di assegnamento nel codice).

Si assuma che la variabile *EXPO* sia stata popolata con n_pad padiglioni (numero intero, positivo e minore di *MAX_PAD*) e che i padiglioni contengano anche la lista dei visitatori in coda. Si assuma inoltre che sia stata dichiarata la variabile *listaVis*, che rappresenta la lista di visitatori di EXPO in un certo giorno che hanno pagato il biglietto e indicato i padiglioni preferiti. Si scrivano porzioni di codice per risolvere le seguenti richieste, ricordandosi di dichiarare tutte le variabili che si ritiene necessario utilizzare.

B. 1) Si acquisisca da tastiera il cognome di un visitatore (per semplicità assumiamo che non ci siano omonimi)

2) Si cerchino in *listaVis* le informazioni relative a quel visitatore e si stampi a video l'elenco dei padiglioni che il visitatore preferisce visitare e il tempo di attesa per ciascuna visita. (NB: Il tempo di attesa a ciascun padiglione si misura assumendo che i visitatori in coda entrino uno alla volta nel padiglione e sapendo che il tempo medio di visita impiegato da ciascun visitatore è quello definito nei campi *ore_visita* e *minuti_visita* del tipo *Padiglione*; ad esempio se ci fossero tre visitatori in coda al padiglione Cile, e se il tempo di visita di questo stesso padiglione fosse 1h e 20min, il tempo di attesa sarebbe uguale a (1h e 20min) * 3 = 240 min.)

C. Si definisca un tipo Bagno per contenere le seguenti informazioni:
   - Tipologia di servizio (può essere solo: pubblico, privato, riservato al personale)
   - costo servizio
   - numero di lavandini
   - se ha prese elettriche a disposizione dei visitatori

   Si modifichi quindi la dichiarazione del tipo Padiglione per associarvi il numero (al più 20) e la lista di tutti i bagni presenti nel padiglione.

Soluzione

A) 1)

```
Padiglione EXPO[MAX_PAD];
int n_pad = 0;
int i;

strcpy(EXPO[0].nome,"Giappone");
EXPO[0].ident = giappone;
EXPO[0].ore_visita = 4;
EXPO[0].minuti_visita = 25;
n_pad++;
```

2)

```
Visitatore primo;

strcpy(primo.nome, "Mario");
strcpy(primo.cognome, "Rossi");

primo.preferiti[0] = italia;
primo.preferiti[1] = giappone;
primo.preferiti[2] = emiratiarabi;
primo.prezzo = 25.0;
```

3)

```
EXPO[0].len_coda = 0;
EXPO[0].coda[EXPO[0].len_coda] = primo;
EXPO[0].len_coda++;
```

B)

```
#define MAX_VIS_NEL_GIORNO 250000 /* numero massimo di visitatori per giorno */

typedef enum {falso, vero} Booleano;
```

```
Visitatori listaVis[MAX_VIS_NEL_GIORNO];
Stringa cogn;
int i, j, k, tempoAttesa;
Booleano trovatoVis, trovatoPad; /* servono per fermare al momento opportuno i cicli di ricerca
                                    /* del visitatore e del padiglione */

scanf("%s", cogn);
trovatoVis = falso;
for(i = 0; i < MAX_VIS_NEL_GIORNO && trovatoVis == falso; i++)
  if(strcmp(listaVis[i].cognome, cogn) == 0)
{
    trovatoVis = vero;
    for(j = 0; j < MAX_PREF; j++)
    {
        trovatoPad = falso;
        for(k = 0; k < MAX_PAD && trovatoPad == falso; k++)
          if(EXPO[k].iden == listaVis[i].preferiti[j])
          {
            trovatoPad = vero;
            tempoAttesa = (EXPO[k].ore_visita * 60 + EXPO[k].minuti_visita) * EXPO[k].len_coda;
            printf("Padiglione %s, tempo di attesa %d\n", EXPO[k].nome, tempoAttesa);
          }
    }
}
```

C)

```
#define MAX_BAGNI 20

typedef enum {pubblico, privato, personale} Tipo;
typedef enum {vero, falso} Boolean;

typedef struct{
        Tipo tipo;
        float prezzo_utilizzo;
        int n_lavandini;
        Boolean prese_elettriche;
} Bagno;

typedef struct {
    Stringa nome;
    Nazione ident;
    int ore_visita, minuti_visita;
    Visitatore coda[MAX_VIS];
    int len_coda;
    Bagno bagni[MAX_BAGNI];
    int n_bagni;
} Padiglione;
```

Esercizio 2 (6 punti)

Scrivendone prima lo pseudo codice, si sviluppi un programma in linguaggio C che svolga le seguenti operazioni:

1. **Definisca un tipo matrice** di interi di dimensione *MAX_R* e *MAX_C*, con *MAX_R* e *MAX_C* definite come costanti
2. **Dichiari una variabile *m*** del tipo matrice definito e tutte le altre variabili necessarie ai fini dell'esercizio
3. **Acquisisca da tastiera i valori** da assegnare alle celle della matrice *m*
4. **Chieda all'utente di fornire il numero della riga** su cui effettuare l'operazione al punto seguente
5. **Calcoli il minimo** tra gli elementi contenuti nella matrice, nella **riga** indicata dall'utente
6. **Copi, per colonne, all'interno di un vettore** (monodimensionale) di dimensioni adeguate **tutti i dati contenuti nella matrice *m*** (in questa fase il programma dovrà inserire nel vettore prima i valori contenuti nella prima colonna della matrice, poi di seguito quelli contenuti nella seconda colonna, e così via).
7. **Esegua la stessa operazione indicata al punto 5** (calcolo del minimo dei valori nella riga indicata della matrice) **lavorando** però **sul vettore** invece che sulla matrice.
8. **Controlli** che il valore calcolato al punto 7 e quello calcolato al punto 5 siano uguali. **In caso affermativo**, stampi "*ok, il valore del minimo e*`" seguito dal valore calcolato. **In caso negativo**, stampi "*errore!*" seguito dai due valori calcolati.

Soluzione

```
/* Pseudo codice
      Inclusione librerie
      Definizione dimensioni matrice
      Definizione tipo matrice e vettore
      Dichiarazione di variabili
      Lettura dati da tastiera con controllo input
         Lettura interi e inserimento in matrice
         Lettura numero riga matrice con controllo input
      Calcolo del minimo tra i valori della riga indicata da tastiera
      Copia nel vettore i valori delle colonne della matrice
      Calcolo del minimo tra i valori nel vettore corrispondenti alla riga della matrice indicata da tastiera
      Controllo equivalenza valori minimi calcolati
         Se uguali, stampa a video "ok, il valore calcolato e` " seguito da valore minimo1 o minimo2
         Se diversi, stampa a video "errore! " seguito da minimo1 e minimo2.
      Fine
*/
#include <stdio.h>

/* Si fa l'ipotesi che la matrice sia di dimensioni 3x5 */
#define MAX_R 3
#define MAX_C 5

typedef int matrice[MAX_R][MAX_C];
typedef int vettore[MAX_R * MAX_C];

int main()
{
```

```
  matrice m;
  vettore v;
  int i, j, k, riga;
  float minimo1, minimo2;

  printf("Inserisci i valori della matrice %dx%d: ", MAX_R, MAX_C);
  for(i = 0; i<MAX_R; i++)
    for(j = 0; j < MAX_C;j++)
      scanf("%d", &m[i][j]);

  printf("Inserisci il numero della riga: ");
  scanf("%d", &riga);
  while((riga >= MAX_R) || (riga < 0))
    scanf("%d", &riga);

  minimo1 = m[riga][0];
  for(i = 1; i < MAX_C; i++){
    if(m[riga][i]<minimo1)
          minimo1 = m[riga][i];
  }

  k = 0;
  for(j = 0; j < MAX_C; j++)
    for(i = 0; i < MAX_R; i++)
    {
      v[k] = m[i][j];
      k = k+1;
    }

  minimo2 = v[riga];
  for(k = riga + MAX_R; k < MAX_R * MAX_C; k = k + MAX_R){
    if(v[k] < minimo2)
          minimo2 = v[k];
  }

  if(minimo1 == minimo2)
    printf("ok, il valore calcolato e` %f\n", minimo1);
  else printf("errore! %f %f\n", minimo1, minimo2);
  return(0);
}
```

Esercizio 3 (4 punti)
Si codifichi in IEEE (754-1975) a 32 bit i seguenti numeri

- 1.4
- 1.5
- 2.95

Indicando quali di questi ha una rappresentazione esatta.

Domanda extra:
Si consideri il seguente frammento di programma. La sua esecuzione porta a stampare
"La somma di 1.4 e 1.5 non è uguale a 2.95"
Si giustifichi brevemente come mai avviene questo.

```c
#include <stdio.h>

int main() {
  float dato1, dato2;

  dato1 = 1.4;
  dato2 = 1.5;

  printf("\n\nLa somma di %f + %f ", dato1, dato2);
  if (dato1+dato2 != 2.90) printf("non ");
  printf("e' uguale a %f\n\n", 2.90);

  return 0;
}
```

Soluzione:

1.4 in binario con codifica IEEE 754-1975 a 32 bit è pari a:

- Segno: 0
- Mantissa: 1.01100110011001100110011 e non necessita di essere normalizzata ed è periodica
- Esponente: 127 -> 01111111

1.5 in binario con codifica IEEE 754-1975 a 32 bit è pari a:

- Segno: 0
- Mantissa: 1.10000000000000000000000 e non necessita di essere normalizzata
- Esponente: 127 -> 01111111

2.9 in binario con codifica IEEE 754-1975 a 32 bit è pari a:

- Segno: 0
- Mantissa: 10.11100110011001100110011 = 1.01110011001100110011001 x 2^1
- Esponente: 127 + 1 = 128 -> 10000000

Per sommare i due numeri dovremo semplicemente sommare le mantisse, dal momento che l'esponennte è lo stesso. Avremo come mantissa: 1.11100110011001100110011, che non è uguale alla mantissa della rappresentazione di 2.9. Il tutto è dovuto quindi a due arrotondamenti differenti operati su 1.4 e 2.9.