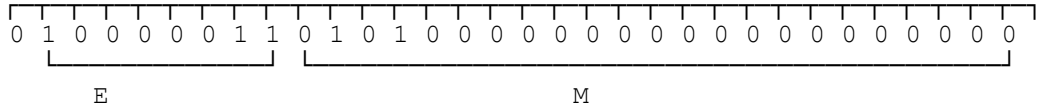
	Politecnico di Milano Facoltà di Ingegneria Industriale **INFORMATICA B** Appello 26 Giugno 2017		COGNOME E NOME
			MATRICOLA
			Spazio riservato ai docenti

- Il presente plico contiene 3 esercizi e **deve essere debitamente compilato con cognome e nome, numero di matricola**.
- Il tempo a disposizione è di 1 ora e 45 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!).
- **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.
- **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**.
- L'esame orale è parte integrante dell'esame e deve essere realizzato almeno sufficientemente per il superamento dell'esame complessivo.

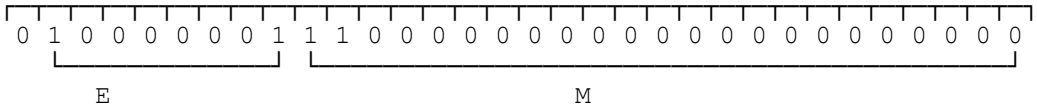
Esercizio 1 (6 punti)

1. Ricavare il numero A in codifica decimale corrispondente alla seguente codifica in virgola mobile:



Suggerimento: a partire dalla conoscenza del processo di conversione da base 10 a base 2, è necessario ragionare a ritroso, ricordando qual è il significato in base 10 di ciascuno degli elementi di cui si compone il numero in base 2.

2. Ricavare il numero B in codifica decimale corrispondente alla seguente codifica in virgola mobile:



3. Esprimere il rapporto A/B nella codifica in virgola mobile IEEE-754 con precisione singola

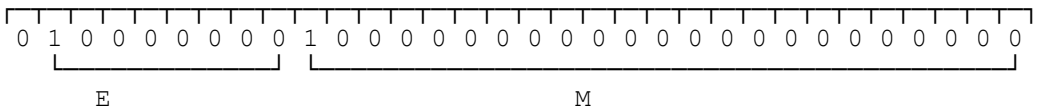
Soluzioni

1 Il numero A in codifica decimale è 21 infatti:
Calcolo dell'esponente in base 10: noi sappiamo che E = esponente di 2 + K, quindi esponente di 2 = E-K. Siccome sono stati usati 8 bit per rappresentare l'esponente, sappiamo di essere nel caso di precisione singola, quindi K = 127. Ricaviamo il valore di E in base 10:
$E_{10} = 2^0+2^1+2^7 = 1+2+128 = 131$
esponente di 2 = E_{10}-K = 131-127 = 4

M è la parte decimale del numero. Il numero complessivo è
1.01010000000000000000000
Per poter eseguire la conversione in base 10, è conveniente spostare la virgola in modo che tutti i valori pari a 1 si trovino a sinistra della virgola:
10101.00000000000000000000
Questo numero sarà ovviamente equivalente al primo solo se moltiplicato per 2^{-4}
10101 in base 10 = $2^0 + 2^2 + 2^4 = 1 + 4 + 16 = 21$
Quindi infine il nostro numero in base 10 è pari a $21 * 2^{-4} * 2^4 = 21$

2 Il numero B in codifica decimale è 7
esponente di 2_{10} = E_{10}-K = $2^0+ 2^7$-127 = 129 - 127 = 2
1.11000000000000000000000 -> 111.00000000000000000000 moltiplicato per 2^{-2} (in base 10)
Quindi infine il nostro numero in base 10 è pari a $7 * 2^{-2} * 2^2 = 7$

3 La codifica di A/B = 21/7 = 3 decimale in virgola mobile IEEE-754 con precisione singola è:



Esercizio 2 (10 punti)

Un'organizzazione che si occupa di archeologia sottomarina vuole sviluppare un software in linguaggio C che permetta di catalogare tutti i reperti archeologici che sono stati scoperti. Ogni reperto è caratterizzato dalle seguenti informazioni:

- coordinate geografiche in cui si trova
- profondità del reperto rispetto al livello del mare
- peso del reperto
- tipo di reperto, che può essere: manufatto in terracotta, manufatto in marmo, manufatto in metallo

Tutti i reperti vengono memorizzati in un array `insiemeReperti` di dimensione 1000.

A: Si definiscano in linguaggio C le strutture dati e le variabili necessarie per rappresentare l'insieme dei reperti nella variabile `insiemeReperti`.

B: Si supponga che la variabile `insiemeReperti` sia stata precedentemente riempita con le informazioni di 1000 reperti e si sviluppi in C il ciclo che, per ogni reperto, stampa a video il suo peso. Si dichiarino anche tutte le variabili necessarie allo scopo che non sono state dichiarate nel punto precedente.

Periodicamente, l'organizzazione progetta spedizioni per raccogliere i reperti scoperti. L'operazione non è sempre possibile perché il veicolo che viene usato per raccogliere i reperti ha una portata massima (un peso massimo che può portare) che non può essere superato. D'altra parte, l'organizzazione vuole fare in modo che il veicolo venga utilizzato nel modo più efficiente possibile e quindi la spedizione verrà pianificata in modo tale che il veicolo raccolga il maggior numero possibile di reperti nell'area prescelta.
La spedizione viene pianificata come segue:
Si parte con un veicolo scarico, di portata `pMax`, da una coppia di coordinate geografiche (`latitudineI` e `longitudineI`) e si esplorano le vicinanze di quelle coordinate geografiche in modo da rimanere nell'ambito della zona di ricerca, i cui confini sono definiti dai punti `latitudineI` ± `soglia`, `longitudineI` ± `soglia`. Per ogni reperto che si trova nei confini definiti, si pianifica di caricare il reperto a bordo del veicolo, a patto che il suo peso, sommato al peso del carico corrente portato dal veicolo, sia minore di `pMax`. In caso contrario, quel reperto viene scartato e si procede a cercarne un altro, sempre rimanendo nei confini dell'area di ricerca. La ricerca termina quando il peso dei reperti da trasportare avrà raggiunto il valore `pMax` oppure quando tutti i reperti nella zona di ricerca saranno stati esaminati.
L'algoritmo di pianificazione del viaggio, durante l'analisi sopra descritta, crea un array di reperti chiamato `caricoVeicolo` che contiene, alla fine dell'analisi, tutti i reperti che dovranno essere caricati sul veicolo durante l'esecuzione della spedizione.

C: Dato un reperto che assumiamo si trovi nella cella i-esima (con i che ha un valore qualsiasi compreso tra 0 e 999, estremi inclusi) dell'array `insiemeReperti`, si scriva la condizione che, se vera, ci permette di dire che il reperto si trova nell'area di ricerca definita dai confini indicati sopra.

D: Si sviluppi il frammento di codice in C necessario per acquisire da tastiera: la portata massima del veicolo, le coordinate geografiche di partenza della spedizione, il valore soglia che definisce i confini di ricerca. Si dichiarino tutte le variabili necessarie allo scopo.

E: Si sviluppi in linguaggio C l'algoritmo di pianificazione del carico descritto in precedenza. Oltre alle dovute inizializzazioni di variabili (non è necessario inizializzare `insiemeReperti` che si assume essere già riempito), la parte principale di questo algoritmo sarà costituita a un ciclo che analizza tutti i reperti in `insiemeReperti` e, per ogni reperto:

- Valuta se si trova nell'area di ricerca (si faccia riferimento alla soluzione fornita per il punto C)
- In caso positivo, valuta se può essere caricato sul veicolo

- In caso positivo, aggiorna la variabile `caricoVeicolo` in modo che contenga i dati del reperto in questione

Si dichiarino anche tutte le variabili necessarie allo scopo che non sono state dichiarate nei punti precedenti.

Soluzione

A)

```
typedef enum {mTerracotta, mMarmo, mMetallo} TipoManufatto;

typedef struct {
   float latitudine;
   float longitudine;
   float profondita;
   float peso;
   TipoManufatto t;
} Reperto;

Reperto insiemeReperti[1000];
```

B)

```
  int i;
  for(i=0; i<1000; i++)
    printf("Peso reperto n. %d: %f\n", i, insiemeReperti[i].peso);
```

C)

```
     insiemeReperti[i].latitudine <= latitudineI+soglia &&
     insiemeReperti[i].latitudine >= latitudineI-soglia &&
     insiemeReperti[i].longitudine <= longitudineI+soglia &&
     insiemeReperti[i].longitudine >= longitudineI-soglia
```

D)

```
  float pMax, latitudineI, longitudineI, soglia;

  printf("Inserisci la portata massima del veicolo: ");
  scanf("%f", &pMax);
  printf("Inserisci la latitudine e longitudine da cui iniziare la ricerca: ");
  scanf("%f%f", &latitudineI, &longitudineI);
  printf("Inserisci il valore della soglia che delimita l'area di ricerca: ");
  scanf("%f", &soglia);
```

E)

```
  int indiceCarico;
  Reperto caricoVeicolo[1000];
  float caricoCorrente;

  /* il veicolo e` vuoto inizialmente*/
  indiceCarico = 0;
```

```
caricoCorrente = 0;
/* scandisco l'insieme dei reperti. La scansione puo` terminare in anticipo se il
   veicolo e` pieno */
for(i=0; i<1000 && caricoCorrente < pMax; i++)
   if (insiemeReperti[i].latitudine <= latitudineI + soglia &&
       insiemeReperti[i].latitudine >= latitudineI - soglia &&
       insiemeReperti[i].longitudine <= longitudineI + soglia &&
       insiemeReperti[i].longitudine >= longitudineI - soglia)

       if(insiemeReperti[i].peso + caricoCorrente <= pMax)
       {
          caricoCorrente = insiemeReperti[i].peso + caricoCorrente;
          caricoVeicolo[indiceCarico] = insiemeReperti[i];
          indiceCarico = indiceCarico + 1;
       }
```

Esercizio 3 (10 punti)

A. Si scriva in linguaggio `MATLAB` una funzione `checkMatrice` che riceva come parametri: una matrice, un numero di riga, un numero di colonna e un valore di soglia. La funzione restituisca `true` se tutti i valori delle celle della matrice nell'intorno della cella con riga e colonna date sono minori del valore di soglia. Le celle nell'intorno della cella identificata dalla riga e dalla colonna date sono quelle immediatamente adiacenti a questa cella (inclusa la cella stessa).
B. Si scriva uno script `MATLAB` che acquisisca da un file di testo, `"matrice"`, i valori di una matrice e da tastiera un numero di riga, un numero di colonna e un valore di soglia, e chiami la funzione `checkMatrice` passandole questi parametri. Si stampi quindi a schermo la scritta "successo!" se il risultato della funzione è vero, "fallimento…" in caso contrario.
C. Si rappresentino *l'ambiente di esecuzione* (workspace) del chiamante (lo script) e del chiamato (la funzione `checkMatrice`) subito prima del momento in cui la funzione termina. Si considerino come valori di matrice, riga, colonna e soglia quelli mostrati qui sotto. Si indichi il valore che la funzione restituirà al chiamante e, conseguentemente, se lo script stamperà "successo!" o "fallimento…" giustificando la risposta:

Caso 1

```
matrice =
     1     2     3     4
     4     3     2     1
     4     3     2     1
riga = 2
colonna = 2
soglia = 5
```

Caso 2

```
matrice =
     1     2     3     4
     4     3     2     1
     4     3     2     1
riga = 2
colonna = 1
soglia = 4
```

Soluzione

A)

```
function [ ris ] = checkMatrice( m, r, c, soglia )
%Restituisce true se tutti i valori nell'intorno di m(r, c) sono minori di
%soglia
%   m: matrice
%   r: numero di riga
%   c: numero di colonna
%   soglia: valore di cui tutti gli elementi nell'intorno di m(r,c) devono
%   essere minori.

 [nrighe, ncolonne] = size(m);
 if (r-1 >= 1)
     if(r+1 <= nrighe)
        if(c-1 >= 1)
            if(c+1 <= ncolonne)
               val = m(r-1:r+1, c-1:c+1)<soglia;
            else val = m(r-1:r+1, c-1:c)<soglia;
            end
```

```
        elseif(c+1 <= ncolonne)
                val = m(r-1:r+1, c:c+1)<soglia;
        else val = m(r-1:r+1, c)<soglia;
        end
    elseif(c-1 >= 1)
        if(c+1 <= ncolonne)
            val = m(r-1:r, c-1:c+1)<soglia;
        else val = m(r-1:r, c-1:c)<soglia;
        end
    elseif(c+1 <= ncolonne)
        val = m(r-1:r, c:c+1)<soglia;
    else val = m(r-1:r, c)<soglia;
    end
 elseif(r+1 <= nrighe)
    if(c-1 >= 1)
        if(c+1 <= ncolonne)
            val = m(r:r+1, c-1:c+1)<soglia;
        else val = m(r:r+1, c-1:c)<soglia;
        end
    elseif(c+1 <= ncolonne)
        val = m(r:r+1, c:c+1)<soglia;
    else val = m(r:r+1, c)<soglia;
    end
 elseif(c-1 >= 1)
    if(c+1 <= ncolonne)
        val = m(r, c-1:c+1)<soglia;
    else val = m(r, c-1:c)<soglia;
    end
 elseif(c+1 <= ncolonne)
    val = m(r, c:c+1)<soglia;
 else val = m(r, c)<soglia;
 end
 ris = all(all(val));

end
```

Abis)

```
function [ ris ] = checkMatrice( m, r, c, soglia )
    [nrighe, ncolonne] = size(m);
    ulr = max(1, r-1);
    ulc = max(1, c-1);
    lrr = min(nrighe, r+1);
    lrc = min(ncolonne, c+1);
    ris = all(all(m(ulr:lrr, ulc:lrc) < soglia));
```

B)

```
load('matrice');
riga = input('inserisci il numero di riga ');
colonna = input('inserisci il numero di colonna ');
soglia = input('inserisci il valore di soglia ');
if checkMatrice(matrice, riga, colonna, soglia)
    disp('successo!')
else disp('fallimento...');
end
```

C)

Ambienti di esecuzione relativi al caso 1

```
matrice =
   1   2   3   4
   4   3   2   1
   4   3   2   1
riga = 2
colonna = 2
soglia = 5
```

Ambiente di esecuzione (workspace) dello script

```
m =
   1   2   3   4
   4   3   2   1
   4   3   2   1
r = 2
c = 2
soglia = 5
nrighe = 3
ncolonne = 4
val =
 3×3 logical array
  1   1   1
  1   1   1
  1   1   1

ris = 1
```

Ambiente di esecuzione (workspace) di checkMatrice

In questo caso, siccome il valore restituito dalla funzione è 1, lo script stampa la stringa "successo!"

Ambienti di esecuzione relativi al caso 2

```
matrice =
   1   2   3   4
   4   3   2   1
   4   3   2   1
riga = 2
colonna = 1
soglia = 4
```

Ambiente di esecuzione (workspace) dello script

```
m =
   1   2   3   4
   4   3   2   1
   4   3   2   1
r = 2
c = 1
soglia = 4
nrighe = 3
ncolonne = 4
val =
 3×2 logical array
  1   1
  0   1
  0   1

ris = 0
```

Ambiente di esecuzione (workspace) di checkMatrice

In questo caso, siccome il valore restituito dalla funzione è 0, lo script stampa la stringa "fallimento...". Si noti che in questo caso la struttura della variabile val in checkMatrice è diversa da quella della stessa variabile nel caso 1. Il motivo sta nel fatto che in questo secondo caso il valore da cui partire per l'analisi della matrice si trova sul bordo della matrice stessa (sulla prima colonna).