Soluzione

1)

```
float bolletta[100];
int i, j, ora;

for (i = 0; i < 100; i++)
{
     bolletta[i] = 0;
     for (j = 0; j < db[i].n; j++)
     {
          ora = db[i].c[j].ora;
          if (ora >= 22 || ora < 8)
               bolletta[i] += db[i].c[j].durata * 0.005;
          else
               bolletta[i] += db[i].c[j].durata * 0.01;
     }
}
```

2)

```
cliente premium[100];
int i, k=0;

for (i = 0; i < 100; i++)
     if (bolletta[i] > 100)
        premium[k++] = clienti[i];
```

Esercizio 3 (3 punti). Le seguenti dichiarazioni definiscono tipi di dati relativi alla categoria degli impiegati di un'azienda (gli impiegati possono essere di prima, seconda, …, quinta categoria), agli uffici occupati da tali impiegati, all'edificio che ospita tali uffici (l'edificio è diviso in 20 piani ognuno con 40 uffici).

```
/* definizioni dei tipi */
typedef struct { char nome[20], cognome[20];
                 int cat;  /* contiene valori compresi tra 1 e 5 */
                 int stipendio;
               } impiegato;
typedef enum {nord, nordEst, est, sudEst, sud, sudOvest, ovest, nordOvest} esposizione;
typedef struct { int superficie;  /* in metri quadri */
                 esposizione esp;
                 impiegato occupante;
               } ufficio;

/* definizioni delle variabili */
ufficio torre[20][40]; /* rappresenta un edificio di 20 piani con 40 uffici per piano */
```

Si scriva un frammento di codice, che includa eventualmente anche le dichiarazioni di ulteriori variabili, che, per tutte e sole le persone che occupano un ufficio (tra quelli memorizzati nella variabile torre) orientato a sud oppure a sudEst e avente una superficie compresa tra 20 e 30 metri quadri, stampi il cognome, lo stipendio e la categoria.

```
int p, u;  /* indice di piano nell'edificio e di ufficio nel piano */

for (p=0; p<20; p++)
   for (u=0; u<40; u++)
      if ( ( torre[p][u].esp==sudEst || torre[p][u].esp==sud) &&
               (torre[p][u].superficie>=20 && torre[p][u].superficie<=30)  )    {
            printf("\n il Signor %s è impiegato di categoria %d",
torre[p][u].occupante.cognome,
torre[p][u].occupante.cat);
            printf (" e ha uno stipendio pari a %d euro \n",
torre[p][u].occupante.stipendio);
      }
```

Esercizio 2 (2 punti). Trasformare il frammento di codice incluso nella parte sinistra della tabella in un equivalente frammento, da scrivere nella colonna destra, che contenga istruzioni while invece che for. Specificare, infine, qual è il valore stampato quando il codice viene eseguito, giustificando adeguatamente la risposta.

```
…
int i, j, s;
s = 1;
for (i=3; i>=0; i--)
   for (j=3; j>=0; j--)
      if ( (i+j) % 2 != 0 )
         s = s * 2;
printf("%d", s);
…
```

```
…
int i, j, s;
s = 1;
i=3;
while (i>=0) {
   j=3;
   while (j>=0) {
      if ( (i+j) % 2 != 0 )
         s = s * 2;
      j--;
  }
  i--;
}
printf("%d", s);
…
```

Il valore stampato è 2^8=256, perchè i due cicli considerano tutte le coppie (i,j) con $0 \le i,j \le 3$, che sono 16 (4*4) ed esattamente 8 di queste hanno somma dispari, quindi il numero *s*, inizialmente pari a 1, viene moltiplicato per 2 esattamente 8 volte.

Esercizio 3 (6 punti)

Si consideri il seguente programma in linguaggio C:

```
#define N -17

int main() {
   int a = N/10;
   float b = N/10.0;
   double c = N/10.0;
   double d = b;
}
```

Sapendo che i tipi delle variabili utilizzate sono codificati in binario in questo modo:

- `int`: complemento a due a 32 bit
- `float`: virgola mobile a precisione singola secondo lo standard IEEE 754-1985 (1 bit per il segno, 23 bit per la mantissa, 8 per l'esponente (K = 127))
- `double`: virgola mobile a precisione doppia secondo lo standard IEEE 754-1985 (1 bit per il segno, 52 bit per la mantissa, 11 per l'esponente (K = 1023))

Si risponda alle seguenti domande:

1. Qual è il valore in decimale e in binario della variabile **a** alla fine dell'esecuzione dell'ultima istruzione? Giustificare la risposta mostrando i calcoli effettuati.
2. Qual è il valore in decimale e in binario della variabile **b** alla fine dell'esecuzione dell'ultima istruzione? Giustificare la risposta mostrando i calcoli effettuati.
3. Considerando il risultato ottenuto nel punto 2, alla fine dell'esecuzione dell'ultima istruzione le variabili **c** e **d** contengono lo stesso valore binario? La risposta cambierebbe se la costante **N** fosse definita pari a 10 anziché a -17? Non è necessario ricavare il valore in binario di **c** e **d**, ma la risposta deve essere adeguatamente motivata.

Soluzione

1. Valore di a in decimale: a = -17/10 = -1. Il risultato si tronca perché è intero. Rappresentazione binaria di 1 come numero naturale a 32 bit: 00000000000000000000000000000001, in complemento a due: -1 è 11111111111111111111111111111110 + 1 = 11111111111111111111111111111111
   Quindi il valore di a in binario è: 11111111111111111111111111111111

2. Valore di a in decimale: b = -17/10 = -1.7
   S = 1 (segno negativo)
   parte intera del numero: 1
   determino la parte frazionaria del numero:
   0.7 x 2 -> 1
   1.4 x 2 -> 0 *
   0.8 x 2 -> 1 *
   1.6 x 2 -> 1 *
   1.2 x 2 -> 0 *
   0.4 ... (il periodo e' 0110)
   parte frazionaria del numero: .10110
   M = 1.10110 x 2^0 (la mantissa è già normalizzata) = (sottintendendo 1. e troncando il numero al 23esimo bit della precisione singola) = 10110011001100110011001
   E = 127 + 0 = 127_{10} = 01111111_2. Quindi il valore di b in binario è:
   10111111110110011001100110011001

3. Dall'analisi dei calcoli effettuati nel punto 2 emerge che la rappresentazione del valore -1.7 in binario è periodica. Questo significa che tale valore sarà diverso a seconda che venga calcolato in precisione singola o doppia. Entrambe le variabili **c** e **d** sono double e consentono quindi di rappresentare numeri in precisione doppia. Alla variabile **d** però viene assegnato il valore di **b** che è già troncato in precisione singola. Per questo motivo, **c** e **d** avranno valori diversi. Per **N = 10** **c** e **d** hanno lo stesso valore in binario perché il valore 1 (10/10) nella codifica binaria in virgola mobile non è periodico e non viene approssimato né in precisione singola né in precisione doppia.

Esercizio 1 (10 punti)

Una fattoria gestisce una stalla con 10 mucche, ciascuna identificata da un codice alfanumerico di 8 caratteri.

Il gestore della fattoria vuole realizzare un programma in C che consenta la contabilizzazione del latte prodotto in un giorno dalle 10 mucche.

In questo contesto, si risponda ai seguenti punti:

1. Si definisca in C il tipo **ProduzioneGiornaliera** che rappresenti la quantità di latte prodotta da ognuna delle 10 mucche della stalla. La produzione giornaliera di ogni mucca deve essere associata al suo codice identificativo. Si definiscano quindi ulteriori tipi se necessario. La quantità di latte prodotta da ciascuna mucca è espressa in litri.

2. Si definisca la variabile **produzione**, utilizzando in modo appropriato il tipo **ProduzioneGiornaliera** definito al punto 1, per la memorizzazione della produzione giornaliera delle 10 mucche della stalla. Si supponga inoltre che la variabile **produzione** sia stata opportunamente inizializzata con la produzione giornaliera delle 10 mucche della stalla.
   Si scriva un frammento di programma C che stampi a video il codice identificativo e la quantità di latte prodotta da ciascuna mucca della stalla. Si stampi inoltre la quantità totale di latte prodotto nella stalla.

SOLUZIONE

```
typedef char stringa[8];

typedef struct {
        stringa codiceMucca;
        float litri;
} ProduzioneMucca;

typedef ProduzioneMucca ProduzioneGiornaliera[10];

ProduzioneGiornaliera produzione;

float tot = 0;

for (i  = 0; i <10 ; i++)
        {
        tot += produzione[i].litri;
        printf("\nla mucca %s ha prodotto %f litri", produzione[i].codiceMucca, produzione[i].litri);
        }

printf("\n\tProduzione totale %f", tot);
```

Esercizio 1 (7 punti)

Siano date le seguenti definizioni di strutture dati:

```
#define MAXDB 80

typedef struct
{
  int giorno;
  int mese;
  int anno;
} data;

typedef char stringa[20];

typedef struct
{
  data      data_di_nascita;
  int       giorno_dell_anno;
  int       codice_crimine;
  stringa   stringa_crimine;
} arresto;

arresto database[MAXDB];
int arresti_per_giorno[365];
```

ove `database` contiene gli arresti effettuati in un determinato anno mentre `arresti_per_giorno` è una variabile array ausiliaria (i cui elementi sono inizializzati tutti a `0`) utilizzabile per i calcoli intermedi.

Scrivere una porzione di codice C che ricavi e stampi a video il massimo numero di arresti giornalieri dei nati nel 1979 con `codice_crimine` pari a `555` utilizzando i dati contenuti in `database`. Si definiscano e inizializzino le eventuali altre variabili temporanee necessarie per tale calcolo. Si assuma che il database **non sia ordinato**.

Soluzione

```
int i;
int max;

for(i=0; i<MAXDB; i++)
  if(database[i].data_di_nascita.anno == 1979 &&
    database[i].codice_crimine == 555)
    arresti_per_giorno[database[i].giorno_dell_anno] += 1;

max = 0;
for(i=0; i<365; i++)
  if(arresti_per_giorno[i] > max)
    max = arresti_per_giorno[i];

printf("Massimo numero di arresti giornalieri: %d", max);
```