d) Una struttura **student** che contenga i vettori di caratteri **firstName[15]** e **lastName[15]**, e la variabile **homeAddress** del tipo **struct address** definito nella parte (c).

e) La struttura **test** che contenga 16 campi di bit di lunghezza 1. I nomi dei campi di bit sono le lettere comprese tra **a** e **p**.

10.5 Date le seguenti definizioni di strutture e dichiarazioni di variabili,

```
struct customer {
   char lastName[15];
   char firstName[15];
   int customerNumber;

   struct {
      char phoneNumber[11];
      char address[50];
      char city[15];
      char state[3];
      char zipCode[6];
   } personal;

} customerRecord, *customerPtr;

customerPtr = &customerRecord;
```

scrivete un'espressione separata che possa essere usata per accedere ai membri della struttura in ognuna delle seguenti parti.

a) Membro **lastName** della struttura **customerRecord**.
b) Membro **lastName** della struttura puntata da **customerPtr**.
c) Membro **firstName** della struttura **customerRecord**.
d) Membro **firstName** della struttura puntata da **customerPtr**.
e) Membro **customerNumber** della struttura **customerRecord**.
f) Membro **customerNumber** della struttura puntata da **customerPtr**.
g) Membro **phoneNumber** del membro **personal** della struttura **customerRecord**.
h) Membro **phoneNumber** del membro **personal** della struttura puntata da **customerPtr**.
i) Membro **address** del membro **personal** della struttura **customerRecord**.
j) Membro **address** del membro **personal** della struttura puntata da **customerPtr**.
k) Membro **city** del membro **personal** della struttura **customerRecord**.
l) Membro **city** del membro **personal** della struttura puntata da **customerPtr**.
m) Membro **state** del membro **personal** della struttura **customerRecord**.
n) Membro **state** del membro **personal** della struttura puntata da **customerPtr**.
o) Membro **zipCode** del membro **personal** della struttura **customerRecord**.
p) Membro **zipCode** del membro **personal** della struttura puntata da **customerPtr**.

10.7 Modificate il programma della Figura 10.16, in modo che mescoli le carte mediante l'uso di un mescolatore ad alta efficienza (come mostrato nella Figura 10.3). Visualizzate il mazzo risultante nel formato a due colonne mostrato nella Figura 10.4. Fate precedere ogni carta dal suo colore.

10.8 Create l'unione **integer** con i membri **char c**, **short s**, **int i** e **long l**. Scrivete un programma che accetti in input valori di tipo **char**, **short**, **int** e **long** e li immagazzini nelle variabili di un'unione di tipo **union integer**. Ogni variabile dell'unione dovrà essere visualizzata come un **char**, uno **short**, un **int** e un **long**. I valori saranno sempre stampati correttamente?

10.9 Create l'unione **floatingPoint** con i membri **float f**, **double d** e **long double l**. Scrivete un programma che accetti in input valori di tipo **float**, **double** e **long double**, e li immagazzini nelle

variabili di un'unione di tipo **union floatingPoint**. Ogni variabile dell'unione dovrà essere visualizzata come un **float**, un **double** e un **long double**. I valori saranno sempre stampati correttamente?

10.10 Scrivete un programma che faccia scorrere a destra di 4 bit una variabile intera. Il programma dovrà visualizzare l'intero in bit, prima e dopo l'operazione di scorrimento. Il vostro sistema metterà degli 0 o degli 1 nei bit svuotati?

10.11 Modificate il programma della Figura 10.7, in modo che possa funzionare con interi di 4 byte, qualora il vostro computer usi interi di 4 byte.

10.12 Far scorrere a sinistra di 1 bit un intero **unsigned** è equivalente a moltiplicare il valore per 2. Scrivete la funzione **power2**, che accetti i due argomenti interi **number** e **pow** e calcoli

number * 2^{pow}

Usate l'operatore di scorrimento per calcolare il risultato. Il programma dovrà visualizzare i valori come interi e come bit.

10.13 L'operatore di scorrimento a sinistra può essere usato per comprimere i valori di due caratteri, in una variabile intera senza segno di due byte. Scrivete un programma che accetti in input dalla tastiera due caratteri e li passi alla funzione **packCharacters**. Per comprimere due caratteri in una variabile intera **unsigned**, assegnatele il primo carattere, fatela scorrere a sinistra per 8 posizioni e combinatela con il secondo carattere, usando l'operatore bitwise OR inclusivo. Il programma dovrà visualizzare i caratteri nel loro formato a bit, prima e dopo essere stati compressi nell'intero **unsigned**, per dimostrare che saranno stati compressi correttamente nella variabile **unsigned**.

10.14 Usando gli operatori di scorrimento a destra, AND bitwise e una maschera, scrivete la funzione **unpackCharacters**, che prenda l'intero **unsigned** dell'Esercizio 10.13 e lo decomprima in due caratteri. Per decomprimere una coppia di caratteri da un intero **unsigned** di due byte, combinatelo con la maschera **65280 (11111111 00000000)** e fate scorrere a destra per otto bit il risultato. Assegnate il valore risultante a una variabile **char**. In seguito combinate l'intero **unsigned** con la maschera **255 (00000000 11111111)**. Assegnate il risultato a un'altra variabile **char**. Il programma dovrà visualizzare in bit l'intero **unsigned**, prima della decompressione, e quindi dovrà visualizzare in bit anche i caratteri, per confermare che la loro decompressione sia stata eseguita correttamente.

10.15 Riscrivete il programma dell'Esercizio 10.13 in modo che comprima 4 caratteri, qualora il vostro sistema usi interi di 4 byte.

10.16 Riscrivete la funzione **unpackCharacters** dell'Esercizio 10.14 in modo che decomprima 4 caratteri, qualora il vostro sistema usi interi di 4 byte. Create le maschere necessarie per decomprimere i 4 caratteri, facendo scorrere a sinistra per 8 bit il valore 255 contenuto nella variabile maschera, per 0, 1, 2 o 3 volte (secondo il byte che state decomprimendo).

10.17 Scrivete un programma che inverta l'ordine dei bit di un valore intero senza segno. Il programma dovrà prendere in input dall'utente il valore e richiamare la funzione **reverseBits** per visualizzare i bit in ordine inverso. Visualizzate il valore in bit, prima e dopo che questi siano stati invertiti, per confermare che siano stati invertiti correttamente.

10.18 Modificate la funzione **displayBits** della Figura 10.7 in modo che sia portabile tra sistemi che usano interi di due o quattro byte. Suggerimento: usate l'operatore **sizeof** per determinare la dimensione di un intero su una macchina particolare.

10.19 Il seguente programma usa la funzione **multiple**, per determinare se l'intero immesso dalla tastiera è un multiplo di qualche intero **X**. Esaminate la funzione **multiple** e, quindi, determinate il valore di **X**.