Un file binario “VOLO.DAT” contiene le informazioni relative alle prenotazioni di un volo di linea. Ogni record contiene i dati relativi ad un cliente; in particolare:

nome, **cognome**, **conferma**

dove **nome**, **cognome**, sono stringhe che individuano univocamente ogni cliente (si esclude cioè che esistano clienti omonimi); **conferma** è un intero che può valere 0 oppure 1:

- **se conferma vale 0**, il cliente è in “lista di attesa” per quel volo (cioè è in attesa che si liberi un posto su quel volo);
- **se conferma vale 1**, la prenotazione del cliente è effettiva (cioè il cliente ha il posto assicurato).

Si distingue, quindi, tra clienti prenotati (conferma = 1) e clienti in attesa (conferma = 0). Nel file considerato, l’alternanza tra record relativi a clienti prenotati e clienti in attesa avviene in modo del tutto casuale. All’inizio del file, inoltre, si trova un numero intero che rappresenta il numero di clienti contenuti nel file. Per quello che riguarda la priorità con cui verranno serviti i clienti in attesa, questa rispetta esattamente l’ordine con cui sono stati inseriti in “VOLO.DAT” (cioè il primo record letto corrisponde al più prioritario, l’ultimo al meno prioritario).

Si scriva un programma C che:

1. Legga il file “VOLO.DAT” ed inserisca i dati letti in un vettore **V** di dimensioni opportune.
2. A partire dal vettore **V**, costruisca due vettori:
   - **W1**, che contiene l’elenco dei nominativi (nome, cognome) dei clienti prenotati (cioè con conferma = 1);
   - **W2**, che contiene l’elenco dei nominativi in lista di attesa (cioè con conferma = 0).

   Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all’interno del file “string.h”, che, date due stringhe, copia il contenuto della seconda nella prima.
3. Legga dallo standard input nome e cognome di un cliente prenotato (cioè presente nel vettore), elimini da W1 il cliente specificato ed estragga da W2 un cliente in attesa per inserirlo in W1. A tal scopo, si utilizzi la funzione `strcmp`, dichiarata all’interno del file “string.h”, che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche.
4. Stampi sullo standard output la lista dei clienti in attesa.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  char nome[15], cognome[30];
  int conferma;
} cliente;

typedef struct {
  char nome[15], cognome[30];
} cli;

main() {
  FILE *fp=fopen("VOLO.DAT", "rb");
  int N, i1=0, i2=0, i;
  cliente *V;
  cli *W1, *W2, c;

  fread(&N, sizeof(int), 1, fp);
  V=(cliente *)malloc(N*sizeof(cliente));
  fread(V, sizeof(cliente), N, fp);
  fclose(fp); /* fine domanda 1 */
  W1=(cli *)malloc(N*sizeof(cli));
  W2=(cli *)malloc(N*sizeof(cli));
  for(i=0; i<N; i++)
    if(V[i].conferma) {
      strcpy(W1[i1].nome, V[i].nome);
      strcpy(W1[i1].cognome, V[i].cognome);
      i1++;
    }
    else {
      strcpy(W2[i2].nome, V[i].nome);
      strcpy(W2[i2].cognome, V[i].cognome);
      i2++;
    } /* fine domanda 2 */
  scanf("%s %s", c.nome, c.cognome);
  for(i=0; i<i1; i++)
    if(!strcmp(W1[i].nome, c.nome) &&
      !strcmp(W1[i].cognome, c.cognome)) {
      i2--;
      strcpy(W1[i].nome, W2[i2].nome);
      strcpy(W1[i].cognome, W2[i2].cognome);
      break;
    } /* fine domanda 3 */
  for(i=0; i<i2; i++)
    printf("%s %s\n", W2[i].nome, W2[i].cognome); /* fine domanda 4 */
  free(V);
  free(W1);
  free(W1);
}
```

Un file binario “F1.DAT” contiene le informazioni relative ad un indirizzario. In particolare, ciascun record riporta:

(cognome, nome, via, CAP, città)

dove:

- cognome è una stringa di 30 caratteri (al massimo);
- nome è una stringa di 30 caratteri (al massimo);
- via è una stringa di 30 caratteri (al massimo);
- CAP è un valore intero;
- città è una stringa di 30 caratteri (al massimo).

Sia dato, inoltre, un secondo file di testo F2.TXT che contiene, per ogni riga, una stringa di massimo 30 caratteri che rappresenta un cognome. Si ipotizzi che in F2.TXT non possano esservi cognomi ripetuti.
Si realizzi un programma C che:

1. Carichi i dati contenuti nel file F1.DAT in un vettore V di dimensione opportuna (il numero di indirizzi contenuti nel file deve essere richiesto all’utente).
2. A partire dal vettore V e dal file F2.TXT, crei un vettore W che contenga gli indirizzi contenuti in V per i quali il valore del cognome è presente anche in F2.TXT (per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all’interno del file “string.h”, che, date due stringhe, copia il contenuto della seconda nella prima; per il confronto tra due stringhe si utilizzi la funzione `strcmp`, dichiarata all’interno del file “string.h”, che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche).
3. Legga da tastiera una stringa C di massimo 30 caratteri corrispondente ad un nome di città, e stampi su schermo tutti gli indirizzi presenti in W con valore del campo città uguale a C.
4. Legga da tastiera le informazioni relative ad un nuovo indirizzo e le aggiunga alla fine del file F1.DAT.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  char cognome[30], nome[30], via[30], citta[30];
  int CAP;
} indirizzo;

main() {
  FILE *fp=fopen("F1.DAT", "rb");
  int N, i, NW=0;
  indirizzo *V, *W, I;
  char C[30];

  scanf("%d", &N);
  V=(indirizzo *)malloc(N*sizeof(indirizzo));
  fread(V, sizeof(indirizzo), N, fp);
  fclose(fp); /* fine domanda 1 */
  fp=fopen("F2.TXT", "rt");
  W=(indirizzo *)malloc(N*sizeof(indirizzo));
  while(!feof(fp)) {
    fscanf(fp, "%s", C);
    for(i=0; i<N; i++)
      if(!strcmp(C, V[i].cognome)) {
        strcpy(W[NW].nome, V[i].nome);
        strcpy(W[NW].cognome, V[i].cognome);
        strcpy(W[NW].via, V[i].via);
        strcpy(W[NW].citta, V[i].citta);
        W[NW].CAP=V[i].CAP;
        NW++;
      }
  }
  fclose(fp); /* fine domanda 2 */
  scanf("%s", C);
  for(i=0; i<NW; i++)
    if(!strcmp(W[i].citta, C))
      printf("%s %s %s %s %d\n", W[i].nome, W[i].cognome, W[i].via,
        W[i].citta, W[i].CAP); /* fine domanda 3 */
  fp=fopen("F1.DAT", "ab");
  scanf("%s %s %s %s %d", I.nome, I.cognome, I.via, I.citta, &I.CAP);
  fwrite(I, sizeof(indirizzo), 1, fp);
  fclose(fp); /* fine domanda 4 */
  free(V);
  free(W);
}
```

Un file binario (INDEX.DAT) contiene le informazioni relative all'indice analitico di un libro. Ciascun record del file riporta una stringa ***parola*** ed un numero di pagina ***N***. I record sono memorizzati nel file in ordine casuale. All'inizio del file è memorizzato il numero totale di record contenuti. In particolare, se la stringa contenuta nel campo **parola** inizia con una lettera maiuscola, essa rappresenta il **titolo** di un capitolo del libro, mentre se la prima lettera è minuscola, il record rappresenta un'**occorrenza** della stringa nel testo considerato. Ciascuna parola (intesa come occorrenza) può comparire più volte; ogni titolo compare, invece, una sola volta. Ad esempio, il contenuto del file potrebbe essere:

Europa	10
fiume	15
America	7
fiume	9
città	9
Alpi	27
alpi	28
città	18
…	

Si scriva un programma C che:

1. Crei in memoria centrale un vettore V di dimensioni opportune rappresentante l'indice analitico, che riporti tutti i record di INDEX.DAT (sia titoli che occorrenze).
2. A partire da V, crei un secondo vettore W che contenga soltanto i record di V corrispondenti ai titoli di capitoli (a tal scopo si supponga di disporre della funzione `int maiuscolo(char c)` che restituisce 1 se e solo se il parametro `c` è maiuscolo). Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima.
3. Costruisca un file di testo "PAROLE.TXT" che contenga soltanto le occorrenze (cioè le Parole che non sono titoli).

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  char parola[15];
  int pagina;
} voce;

int maiuscolo(char c);

main() {
  FILE *fp=fopen("INDEX.DAT","rb");
  int N, i, NW=0;
  voce *V, *W;

  fread(&N, sizeof(int), 1, fp);
  V=(voce *)malloc(N*sizeof(voce));
  fread(V, sizeof(voce), N, fp);
  fclose(fp); /* fine domanda 1 */
  W=(voce *)malloc(N*sizeof(voce));
  for(i=0; i<N; i++)
    if(maiuscolo(V[i].parola[0])) {
      strcpy(W[NW].parola, V[i].parola);
      W[NW].N=V[i].N;
      NW++;
    } /* fine domanda 2 */
  fp=fopen("PAROLE.TXT","wt");
  for(i=0; i<N; i++)
    if(!maiuscolo(V[i].parola[0]))
      fprintf(fp, "%s\n", V[i].parola);
  fclose(fp); /* fine domanda 3 */
  free(V);
  free(W);
}
```

Un file di record (SEGNALATI.DAT) contiene informazioni (cognome, nome, data di nascita, luogo di nascita, codice fascicolo) su alcune persone schedate alla questura. All'inizio del file si trova il numero di elementi ivi contenuti. In seguito all'arresto di uno schedato occorre rimuovere il record che lo riguarda dal file e trasferirlo in un secondo file già esistente (ARRESTATI.DAT).
A questo scopo, si scriva un programma C che:

1. Legga gli elementi del file SEGNALATI.DAT e li inserisca in un vettore V di dimensioni opportune. In ciascun elemento del vettore si preveda un campo aggiuntivo (arrestato, di tipo booleano), inizialmente con valore falso, che indica se lo schedato è stato arrestato (per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima).
2. Legga a terminale i dati di uno schedato (nome, cognome, data di nascita) che è stato arrestato di recente ed accedendo al vettore V:
   - ne modifichi il campo arrestato;
   - visualizzi a terminale il codice del fascicolo.

   A tal scopo si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche.
3. A partire dal vettore V, riscriva il file SEGNALATI.DAT con le informazioni sui segnalati non arrestati ed aggiunga, invece, al file ARRESTATI.DAT le informazioni che riguardano i segnalati arrestati.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct { int giorno, mese, anno; } data;
typedef struct {
  char cognome[30], nome[15], luogo_n[50];
  data data_n;
  int fascicolo;
} schedato;
typedef struct {
  char cognome[30], nome[15], luogo_n[50];
  data data_n;
  int fascicolo, arrestato;
} arresto;

main() {
  FILE *fp=fopen("SEGNALATI.DAT", "rb"), *fp1;
  int N, i, M; schedato s; arresto *V;

  fread(&N, sizeof(int), 1, fp);
  V=(arresto *)malloc(N*sizeof(arresto));
  for(i=0; i<N; i++) {
    fread(s, sizeof(schedato), 1, fp);
    strcpy(V[i].cognome, s.cognome);
    strcpy(V[i].nome, s.nome);
    strcpy(V[i].luogo_n, s.luogo_n);
    V[i].data_n.giorno=s.data_n.giorno;
    V[i].data_n.mese=s.data_n.mese;
    V[i].data_n.anno=s.data_n.anno;
    V[i].fascicolo=s. fascicolo;
    V[i].arrestato=0;
  }
  fclose(fp); /* fine domanda 1 */
  scanf("%s %s %d %d %d", s.nome, s.cognome, &s.data_n.giorno,
    &s.data_n.mese, &s.data_n.anno);
  M=N;
  for(i=0; ; i++)
    if(!strcmp(V[i].nome, s.nome) && !strcmp(V[i].cognome, s.cognome) &&
      (V[i].data_n.giorno=s.data_n.giorno) &&
      (V[i].data_n.mese=s.data_n.mese) &&
      (V[i].data_n.anno=s.data_n.anno)) {
        V[i].arrestato=1;
        printf("%d\n", V[i].fascicolo);
        M--;
        break;
    } /* fine domanda 2 */
  fp=fopen("SEGNALATI.DAT", "wb");
  fp1=fopen("ARRESTATI.DAT", "ab");
  fwrite(&M, sizeof(int), 1, fp);
  for(i=0; i<N; i++) {
    strcpy(s.cognome, V[i].cognome);
    strcpy(s.nome, V[i].nome);
    strcpy(s.luogo_n, V[i].luogo_n);
    s.data_n.giorno= V[i].data_n.giorno;
    s.data_n.mese= V[i].data_n.mese;
    s.data_n.anno= V[i].data_n.anno;
    if(V[i].arrestato) fwrite(&s, sizeof(schedato), 1, fp1);
    else fwrite(&s, sizeof(schedato), 1, fp);
  fclose(fp); fclose(fp1); /* fine domanda 3 */
  free(V);
}
```

Un file binario (MAGAZZINO.DAT) contiene le informazioni relative agli articoli presenti in un magazzino di vendite all'ingrosso. In particolare, ciascun articolo venduto dal magazzino è rappresentato dai seguenti dati:

- **codice**: un valore numerico che individua univocamente l'articolo
- **descrizione**: una stringa che contiene il nome dell'articolo
- **prezzo**: un intero che rappresenta il prezzo di vendita
- **quantità**: un intero che rappresenta il numero di pezzi presenti in magazzino.

Un secondo file di testo (ACQUISTI.TXT) contiene una lista di articoli acquistati (e quindi da eliminare dal magazzino): ogni linea del file ACQUISTI.TXT rappresenta l'acquisto di una certa quantità di un certo articolo A ed in particolare:

- il codice dell'articolo A acquistato;
- il numero di pezzi di A acquistati.

Si vuole realizzare un programma di gestione del magazzino. A questo scopo, si scriva un programma C che:

1. Legga gli elementi contenuti in MAGAZZINO.DAT e li inserisca in un vettore V opportunamente dimensionato (il numero di articoli contenuti nel file deve essere richiesto all'utente).
2. A partire dal file ACQUISTI.TXT, aggiorni il vettore V decrementando opportunamente il valore del campo **quantità** relativo ad ogni articolo acquistato di un valore pari al numero di pezzi acquistati. A tal scopo si supponga l'esistenza di una funzione `trova_cod` che, dato un vettore di articoli ed un codice, restituisca l'indice corrispondente all'articolo avente come codice quello dato.
3. Stampi il prezzo totale (relativo a tutti gli articoli acquistati nelle quantità specificate in ACQUISTI.TXT).
4. A partire da V, riscriva il file MAGAZZINO.DAT inserendovi soltanto gli articoli non esauriti (cioè i record con quantità maggiore di zero). I codici degli articoli esauriti (cioè con quantità uguale a zero) vengano invece scritti in un terzo file di testo ESAURITI.TXT.
5. Si scriva il codice della funzione `trova_cod`.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  int codice, prezzo, quantita;
  char descr[30];
} articolo;

int trova_cod(articolo V[], int codice);

main() {
  FILE *fp=fopen("MAGAZZINO.DAT", "rb"), *fp1;
  int N, i, cod, q, totale=0;
  articolo *V;

  scanf("%d", &N);
  V=(articolo *)malloc(N*sizeof(articolo));
  fread(V, sizeof(articolo), N, fp);
  fclose(fp); /* fine domanda 1 */
  fp=fopen("ACQUISTI.TXT", "rt");
  while(!feof(fp)) {
    fscanf(fp, "%d %d", &cod, &q);
    i=trova_cod(V, cod);
    V[i].quantita-=q;
    totale+=prezzo*q;
  }
  fclose(fp); /* fine domanda 2 */
  printf("%d", totale); /* fine domanda 3 */
  fp=fopen("MAGAZZINO.DAT", "wb");
  fp1=fopen("ESAURITI.TXT", "wt");
  for(i=0; i<N; i++)
    if(V[i].quantita>0) fwrite(&(V[i]), sizeof(articolo), 1, fp);
    else fprintf(fp1, "%d\n", V[i].codice);
  fclose(fp);
  fclose(fp1); /* fine domanda 4 */
  free(V);
}

int trova_cod(articolo V[], int codice) {
  int i;

  for(i=0; ; i++)
    if(V[i].codice==codice) return i;
} /* fine domanda 5 */
```

Si realizzi un programma C per la gestione dei conti correnti presso una filiale bancaria. In particolare, si consideri un file binario “inizio.dat” che contiene, per ogni conto, l’ammontare iniziale del deposito; quindi, ogni elemento di “inizio.dat” conterrà le seguenti informazioni:

- **numero_conto**: un intero che identifica il conto corrente;
- **valore**: un intero positivo che rappresenta la cifra inizialmente depositata nel conto corrente.

Si supponga, inoltre, che il numero totale di conti sia presente all’inizio del file come valore intero.
Un secondo file binario “movimenti.dat” contiene gli aggiornamenti effettuati sui conti correnti (versamenti o prelievi) a partire dalla situazione iniziale. In particolare, quindi, la struttura del file “movimenti.dat” è uguale alla struttura del file “inizio.dat”:

- **numero_conto**: un intero che individua il conto corrente:
- **valore**: un intero (**positivo o negativo**) che rappresenta l’operazione effettuata sul conto dato: se valore è positivo, si tratta di un versamento, se è negativo, si tratta di un prelievo.

Va notato che, mentre il file “inizio.dat” contiene un solo elemento per ogni conto corrente, il file “movimenti.dat” può contenere un qualunque numero di elementi associati allo stesso conto corrente.
Il programma deve:

1. A partire dal file “inizio.dat”, creare un vettore V, opportunamente dimensionato (a tal scopo si supponga che il numero di elementi contenuti nel file “inizio.dat” debba essere richiesto all’utente), che rappresenti la situazione iniziale.
2. A partire dal file “movimenti.dat”, aggiornare opportunamente gli elementi del vettore V; V conterrà quindi un elemento per ogni conto corrente C nel cui campo **valore** sarà contenuta la somma algebrica di tutti i valori associati al conto C in “inizio.dat” ed in “movimenti.dat”.
3. A partire da V, costruire un secondo vettore W contenente gli elementi di L per i quali il campo valore è **minore o uguale a 0**.
4. Scrivere in un **file di testo** “passivi.txt” i numeri di conto corrente ai quali è associato un **valore** negativo.

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct {
  unsigned int numero, valore;
} conto;

main() {
  FILE *fp=fopen("inizio.dat", "rb");
  int N, NW=0;
  conto *V, c, *W;

  fread(&N, sizeof(int), 1, fp);
  V=(conto *)malloc(N*sizeof(conto));
  fread(V, sizeof(conto), N, fp);
  fclose(fp); /* fine domanda 1 */
  fp=fopen("movimenti.dat", "rb");
  while(!feof(fp)) {
    fread(&c, sizeof(conto), 1, fp);
    for(i=0; ; i++)
      if(V[i].numero==c.numero) {
        V[i].valore-=c.valore;
        break;
      }
  }
  fclose(fp); /* fine domanda 2 */
  W=(conto *)malloc(N*sizeof(conto));
  for(i=0; i<N; i++)
    if(V[i].valore<=0) {
      W[NW].numero=V[i].numero;
      W[NW].valore=V[i].valore;
      NW++;
    } /* fine domanda 3 */
  fp=fopen("passivi.txt", "wt");
  for(i=0; i<NW; i++)
    if(W[i].valore<0) fprintf(fp, "%d\n", W[i].numero);
  fclose(fp); /* fine domanda 4 */
  free(V);
  free(W);
}
```

Si realizzi un programma C per la gestione delle vendite in un magazzino all'ingrosso. Un file binario "vendite.dat" contiene le informazioni relative agli articoli venduti nel mese in corso; in particolare, per ogni vendita sono registrate le seguenti informazioni:

- **codice_articolo**, una stringa alfanumerica che rappresenta univocamente l'articolo venduto;
- **codice_cliente**, una stringa alfanumerica che rappresenta univocamente il cliente che ha effettuato l'acquisto;
- **descrizione**, una stringa contenente la descrizione a parole dell'articolo venduto;
- **quantità**, un intero che rappresenta il numero di pezzi venduti;
- **importo_totale**, un intero che rappresenta l'importo totale della vendita;
- **pagato**, un valore logico che indica se l'importo è stato saldato o meno.

Inoltre, il file contiene, come primo valore, il numero totale di record presenti all'interno del file stesso. Un secondo file binario "clienti.dat" contiene le informazioni relative ai clienti del magazzino; in particolare, per ciascun cliente vengono registrate le seguenti informazioni:

- **codice_cliente**, una stringa alfanumerica che rappresenta univocamente il cliente;
- **cognome, nome, indirizzo**, tre stringhe contenenti il cognome, il cognome e l'indirizzo del cliente.

Il programma deve:

1. A partire dal file "vendite.dat" costruire un vettore V di dimensioni appropriate contenente i dati sulle vendite.
2. A partire da V, costruisca 2 vettori V1 e V2: V1 contiene tutte le vendite il cui importo è già stato saldato, mentre V2 contiene tutte le vendite il cui importo è ancora da saldare (per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima).
3. A partire dal file "clienti.dat" e da V2 costruire un file di testo "debitori.txt" che contenga nome, cognome ed indirizzo dei clienti debitori (cioè dei clienti per i quali esiste almeno un elemento in V2). Per il confronto tra stringhe si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  char articolo[10], cliente[10], descr[30];
  int quantita, importo, pagato;
} vendita;

typedef struct {
  char codice[10], cognome[30], nome[15], indirizzo[50];
} cliente;

main() {
  FILE *fp=fopen("vendite.dat", "rb"), *fp;
  int N, N1=0, N2=0, i;
  vendita *V, *V1, *V2;
  cliente c;

  fread(&N, sizeof(int), 1, fp);
  V=(vendita *)malloc(N*sizeof(vendita));
  fread(V, sizeof(vendita), N, fp);
  fclose(fp); /* fine domanda 1 */
  V1=(vendita *)malloc(N*sizeof(vendita));
  V2=(vendita *)malloc(N*sizeof(vendita));
  for(i=0; i<N; i++)
    if(V[i].pagato) {
      strcpy(V1[N1].articolo, V[i].articolo);
      strcpy(V1[N1].cliente, V[i].cliente);
      strcpy(V1[N1].descr, V[i].descr);
      V1[N1].quantita=V[i].quantita;
      V1[N1].importo=V[i].importo;
      V1[N1].pagato=1;
      N1++;
    }
    else {
      strcpy(V2[N2].articolo, V[i].articolo);
      strcpy(V2[N2].cliente, V[i].cliente);
      strcpy(V2[N2].descr, V[i].descr);
      V2[N2].quantita=V[i].quantita;
      V2[N2].importo=V[i].importo;
      V2[N2].pagato=0;
      N2++;
    } /* fine domanda 2 */
  fp=fopen("clienti.dat", "rb");
  fp1=fopen("debitori.txt", "wt");
  while(!feof(fp)) {
    fread(&c, sizeof(cliente), 1, fp);
    for(i=0; ; i++)
      if(!strcmp(c.codice, V2[i].cliente)) {
        fprintf(fp1, "%s %s %s\n", c.nome, c.cognome, c.indirizzo);
        break;
      }
  }
  fclose(fp);
  fclose(fp1); /* fine domanda 3 */
  free(V);
  free(V1);
  free(V2);
}
```

Si consideri una società che gestisce l'erogazione di Gas e Acqua in una certa città. La società effettua periodicamente un consuntivo dei consumi effettuati dagli utenti. A questo proposito vengono utilizzati **tre file binari** dati: utenti.dat, consumi.gas, consumi.acq. In particolare:

- Il file **utenti.dat** contiene l'archivio di tutti gli utenti della società ed il primo valore contenuto nel file indica il numero di record presenti; la struttura di ciascun record di utenti.dat è del tipo:
  - **codice_cliente**: una stringa alfanumerica che individua in modo univoco il cliente;
  - **nome**: una stringa che contiene il cognome ed il nome e dell'utente;
  - **indirizzo**: una stringa che rappresenta l'indirizzo dell'utente
- Il file **consumi.gas** e **consumi.acq** contengono rispettivamente i dati relativi ai consumi di gas e di acqua; la struttura di ciascun record di consumi.gas e consumi.acq sarà identica:
  - **codice_cliente**: l'identificatore alfanumerico dell'utente;
  - **consumo**: il numero di metri cubi (di gas o di acqua) consumati dal cliente;
  - **prezzo_unitario**: il prezzo per metro cubo (di gas o di acqua).

Il programma deve:

1. A partire dal file utenti.dat, creare un vettore V di dimensioni opportune contenente le informazioni su tutti gli utenti.
2. A partire dal vettore V e dagli altri due file dati (consumi.gas e consumi.acq), costruire un secondo vettore W che contenga i dati relativi alle bollette di ciascun utente (sia di acqua che di gas); la struttura di ciascun elemento del vettore sarà quindi del tipo: **nome**, **indirizzo**, **importo_gas**, **importo_acqua**, dove importo_gas e importo_acqua sono le quote totali da pagare rispettivamente per gas e acqua relative all'utente considerato (ovviamente importo_gas e importo_acqua vengono calcolati moltiplicando il consumo per il prezzo unitario). Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima; per il confronto tra stringhe si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche.
3. A partire dal vettore W costruito al punto precedente, stampare gli incassi totali previsti dalla società rispettivamente per gas e per acqua.
4. Costruire un file di testo **fatture.txt** nel quale ogni riga riporti le informazioni relative a ciò che ogni utente deve pagare; in particolare: **nome**, **indirizzo**, **prezzo totale**, dove il prezzo totale è la somma degli importi relativi al consumo di gas e al consumo di acqua.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct{ char codice[10], nome[50], indirizzo[50]; } utente;
typedef struct {
  char codice[10];
  int cons, prezzo;
} consumo;
typedef struct{
  char nome[50], indirizzo[50];
  int gas, acqua;
} bolletta;

main() {
  FILE *fp=fopen("utenti.dat", "rb");
  int N, i, tot_gas=0, tot_acqua=0;
  utente *V;
  bolletta *W;
  consumo c;

  fread(&N, sizeof(int), 1, fp);
  V=(utente *)malloc(N*sizeof(utente));
  fread(V, sizeof(utente), N, fp);
  fclose(fp); /* fine domanda 1 */
  W=(bolletta *)malloc(N*sizeof(bolletta));
  for(i=0; i<N; i++) {
    strcpy(W[i].nome, V[i].nome);
    strcpy(W[i].indirizzo, V[i].indirizzo);
  }
  fp=fopen("consumi.gas", "rb");
  while(!feof(fp)) {
    fread(&c, sizeof(consumo), 1, fp);
    for(i=0; ; i++)
      if(!strcmp(c.codice, V[i].codice)) {
        W[i].gas=c.cons*c.prezzo;
        break;
      }
  }
  fclose(fp);
  fp=fopen("consumi.acq", "rb");
  while(!feof(fp)) {
    fread(&c, sizeof(consumo), 1, fp);
    for(i=0; ; i++)
      if(!strcmp(c.codice, V[i].codice)) {
        W[i].acqua=c.cons*c.prezzo;
        break;
      }
  }
  fclose(fp); /* fine domanda 2 */
  for(i=0; i<N; i++) {
    tot_gas+=W[i].gas;
    tot_acqua+=W[i].acqua;
  }
  printf("%d %d\n", tot_gas, tot_acqua); /* fine domanda 3 */
  for(i=0; i<N; i++)
    printf("%s %s %d\n", W[i].nome, W[i].indirizzo, W[i].gas+W[i].acqua);
  /* fine domanda 4 */
  free(V);
  free(W);
}
```

Si consideri un villaggio turistico in cui i clienti vengono alloggiati in bungalow. Le prenotazioni nel villaggio turistico possono riferirsi soltanto ad intere settimane (non è cioè possibile prenotare un bungalow per meno di una settimana). Per gestire le prenotazioni relative ad una certa settimana, la direzione del villaggio utilizza due file:

- **bungalow.dat:** un file binario che contiene la lista di tutti i bungalow del villaggio;
- **prenotazioni.dat**: un file binario che contiene la lista delle prenotazioni ricevute per la settimana considerata.

In particolare, ciascun record di **bungalow.dat** è rappresentato dalle seguenti informazioni:

- **numero**: un intero che individua univocamente il bungalow;
- **posti:** un intero che indica il numero di posti letto disponibili nel bungalow;
- **servizi**: un valore "booleano" che indica se il bungalow è dotato del bagno;
- **occupato**: un valore "booleano" che indica se il bungalow è occupato o meno.

Ciascun elemento di **prenotazioni.dat** è invece rappresentato dalle seguenti informazioni:

- **cognome**: una stringa che rappresenta il cognome della persona che ha richiesto la prenotazione;
- **persone:** un intero che indica il numero di posti letto richiesti;
- **servizi:** un valore "booleano" che indica se viene richiesto un bungalow con bagno oppure senza.

Scrivere un programma C che permetta una gestione automatizzata delle prenotazioni. A questo scopo il programma deve:

1. A partire dal file **bungalow.dat**, costruire un vettore V dei bungalow di dimensione opportuna contenente i dati sui bungalow (a tal fine, si supponga che il numero di bungalow contenuti nel file venga richiesto all'utente via input prima della lettura del file stesso).
2. Per ogni elemento contenuto nel file **prenotazioni.dat,** ricercare in V un bungalow non occupato (se esiste) che soddisfi la richiesta del cliente (numero posti e presenza di bagno); a tal scopo si supponga l'esistenza di una funzione `trova` che, dato un vettore di bungalow, un intero che ne rappresenta le dimensioni e la richiesta del cliente, restituisca l'indice corrispondente al primo bungalow in grado di soddisfare tale richiesta (o un numero negativo se la richiesta non può essere soddisfatta). Se il bungalow richiesto è presente in V, occorre modificare opportunamente il record corrispondente e stampare su schermo il nome del cliente ed il numero di bungalow assegnato; altrimenti scrivere in un file binario **rifiutati.dat** i dati (cognome, persone, servizi) relativi alla prenotazione inevasa.
3. Si scriva il codice della funzione `trova`.

```
#include <stdio.h>
#include <stdlib.h>

typedef struct {
  int numero, posti, servizi, occupato;
} bungalow;

typedef struct {
  char cognome[30];
  int posti, servizi;
} richiesta;

int trova(bungalow *V, int posti, servizi);

main() {
  FILE *fp=fopen("bungalow.dat", "rb"), *fp1;
  int N, i;
  bungalow *V;
  richiesta r;

  scanf("%d", &N);
  V=(bungalow *)malloc(N*sizeof(bungalow));
  fread(V, sizeof(bungalow), N, fp);
  fclose(fp); /* fine domanda 1 */
  fp=fopen("prenotazioni.dat", "rb");
  fp1=fopen("rifiutati.dat", "wb");
  while(!feof(fp)) {
    fread(&r, sizeof(richiesta), 1, fp);
    i=trova(V, N, richiesta.posti, richiesta.servizi);
    if(i<0) fwrite(&r, sizeof(richiesta), 1, fp1);
    else {
      V[i].occupato=1;
      printf("%s %d\n", r.cognome, V[i].numero);
    }
  }
  fclose(fp1);
  fclose(fp); /* fine domanda 2 */
  free(V);
}

int trova(bungalow *V, int N, posti, servizi) {
  int i;

  for(i=0; i<N; i++)
    if(!V[i].occupato && (V[i].posti==posti) && (V[i].servizi==servizi)) return i;
  return -1;
} /* fine domanda 3 */
```

In un parco di interesse naturalistico è presente un noleggio di biciclette. Le biciclette possono essere noleggiate (**a ore**) dai visitatori del parco. L'archivio di **tutte** le biciclette gestite dal noleggio è rappresentato da un file binario **bici.dat**; in particolare, ogni record contenuto in bici.dat rappresenta una bicicletta e la sua struttura è costituita dai seguenti campi:

- **codice:** una stringa che individua univocamente la bicicletta;
- **modello:** una stringa che registra il modello della bicicletta;
- **tariffa:** un intero che rappresenta la tariffa oraria relativa al noleggio della bicicletta.

Il numero totale di biciclette contenute nel file è indicato da un numero intero presente all'inizio del file stesso. I dati relativi alle biciclette noleggiate (cioè non presenti presso la sede del noleggio) sono registrati in un secondo file binario **fuori.dat**; la struttura di ogni record di fuori.dat è la seguente:

- **codice**: il codice identificatore della bicicletta;
- **cliente**: il nome della persona che ha preso a noleggio la bicicletta;
- **ora_noleggio**: un intero che indica l'orario di prelievo della bicicletta (si trascurano i minuti).

Si realizzi un programma C che:

1. A partire dal file bici.dat costruisca un vettore V di dimensioni opportune contenente le informazioni sulle biciclette.
2. A partire dal vettore V e dal file fuori.dat costruisca un secondo vettore W che rappresenti la situazione aggiornata del noleggio. In particolare, ogni elemento di W deve essere composto da: codice, modello, tipo, tariffa, cliente, ora_noleggio. A tal scopo si supponga l'esistenza di una funzione `trova_bici` che, dato un vettore di biciclette, ed un codice, restituisca l'indice corrispondente alla bicicletta identificata da tale codice. I campi cliente ed ora_noleggio sono ovviamente significativi soltanto nel caso di biciclette noleggiate; nel caso di biciclette disponibili (e quindi non presenti in fuori.dat) si imponga **ora_noleggio=-1**. Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima.
3. A partire dal vettore W, si realizzi un meccanismo di **restituzione** delle biciclette; in particolare, dati da standard input un **codice** ed un'**ora** (che rappresentano rispettivamente il codice della bicicletta restituita e l'orario di restituzione), il programma deve aggiornare opportunamente il vettore W, in modo da rendere nuovamente disponibile la bicicletta. Si stampino inoltre su schermo le informazioni relative alla fattura: **nome del cliente** e **importo da pagare** (= numero di ore * tariffa).
4. Si scriva il codice della funzione `trova_bici`. Per il confronto tra stringhe si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  char codice[6], modello[20]; int tariffa;
} bici;
typedef struct {
char codice[6], cliente[30]; int ora;
} richiesta;
typedef struct {
  bici b; char cliente[30]; int ora;
} noleggio;

int trova_bici(bici V[], char codice[]);

main() {
  FILE *fp=fopen("bici.dat", "rb");
  int N, i, ora;
  char codice[6];
  bici *V;
  noleggio *W;
  richiesta r;

  fread(&N, sizeof(int), 1, fp);
  V=(bici *)malloc(N*sizeof(bici));
  fread(V, sizeof(bici), N, fp);
  fclose(fp); /* fine domanda 1 */
  W=(noleggio *)malloc(N*sizeof(noleggio));
  for(i=0; i<N; i++) {
    strcpy(W[i].bici.codice, V[i].codice);
    strcpy(W[i].bici.modello, V[i].modello);
    W[i].bici.tariffa=V[i].tariffa;
    strcpy(W[i].cliente, "");
    W[i].ora=-1;
  }
  fp=fopen("fuori.dat", "rb");
  while(!feof(fp)) {
    fread(&r, sizeof(richiesta), 1, fp);
    i=trova_bici(V, r.codice);
    strcpy(W[i].cliente, r.cliente);
    W[i].ora=r.ora;
  }
  fclose(fp); /* fine domanda 2 */
  scanf("%s %d", codice, &ora);
  i=trova_bici(V, codice);
  printf("%s %d\n", W[i].cliente, tariffa*(ora-W[i].ora));
  strcpy(W[i].cliente, "");
  W[i].ora=-1; /* fine domanda 3 */
  free(V);
  free(W);
}

int trova_bici(bici V[], char codice[]) {
  int i;

  for(i=0; ; i++)
    if(!strcmp(V[i].codice, codice)) return i;
} /* fine domanda 4 */
```

Per gestire gli accessi a un laboratorio informatico, vengono utilizzati quotidianamente due file:

- **MACCHINE.DAT**, un file binario che contiene i dati relativi alle macchine presenti nel laboratorio: numero identificativo della macchina e tipo di sistema operativo (appartenente al dominio {MSDOS, Windows95, Unix}); il file contiene, all'inizio, un intero che indica il numero totale di macchine presenti in laboratorio;
- **RICHIESTE.DAT**, un file binario che riporta i dati degli utenti che desiderano accedere al laboratorio nel giorno corrente. Per ciascun utente è specificato il nome e il tipo di sistema operativo richiesto.

Si realizzi un programma C che:

1. A partire dal file **MACCHINE.DAT,** crei inizialmente un vettore V, di dimensioni opportune, contenente le informazioni sulle macchine del laboratorio. Tale vettore deve riportare, per ciascuna macchina, il numero identificativo, il tipo si sistema operativo ed un campo che indica se la macchina è stata assegnata (inizialmente falso). Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima.
2. Per ogni richiesta letta dal file **RICHIESTE.DAT**, accedendo al vettore V, verifichi se tale richiesta può essere soddisfatta. Una richiesta può essere soddisfatta se esiste nel vettore una macchina con il sistema operativo richiesto:
   - **se esiste**, si assegna la macchina alla richiesta modificando il campo che indica se la macchina è stata assegnata (attribuendogli valore vero) e si scrivono i dati dell'utente e della macchina assegnata in un terzo file **PRENOTAZIONI.TXT**, di tipo testo.
   - **se non esiste** una macchina disponibile in grado di soddisfare la richiesta, i dati della richiesta vengono scritti in un file binario **INEVASE.DAT**.

   A tal scopo si supponga l'esistenza di una funzione `trova` che, dato un vettore di macchine, un intero che ne rappresenta la lunghezza ed un tipo di sistema operativo, restituisca l'indice corrispondente alla prima macchina libera dotata del sistema operativo richiesto, o un numero negativo se non esiste alcuna macchina in grado di soddisfare la richiesta.
3. Si scriva il codice della funzione `trova`. Per il confronto tra stringhe si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  int numero;
  char so[10];
} macchina;
typedef struct {
  char nome[30], so[10];
} richiesta;
typedef struct {
  macchina m;
  int assegnata;
} lab;

int trova(lab V[], int N, char so[]);

main() {
  FILE *fp=fopen("MACCHINE.DAT", "rb"), *fp1, *fp2;
  int N, i;
  lab *V;
  macchina m;
  richiesta r;

  fread(&N, sizeof(int), 1, fp);
  V=(lab *)malloc(N*sizeof(lab));
  for(i=0; i<N; i++) {
    fread(&m, sizeof(macchina), 1, fp);
    V[i].m.numero=m.numero;
    strcpy(V[i].m.so, m.so);
    V[i].assegnata=0;
  }
  fclose(fp); /* fine domanda 1 */
  fp=fopen("RICHIESTE.DAT", "rb");
  fp1=fopen("PRENOTAZIONI.TXT", "wt");
  fp2=fopen("INEVASE.DAT", "wb");
  while(!feof(fp)) {
    fread(&r, sizeof(richiesta), 1, fp);
    i=trova(V, N, r.so);
    if(i<0) fwrite(&r, sizeof(richiesta), 1, fp2);
    else {
      V[i].assegnata=1;
      fprintf(fp1, "%s %d %s\n", r.cliente, V[i].numero, r.so);
    }
  }
  fclose(fp1);
  fclose(fp2);
  fclose(fp); /* fine domanda 2 */
  free(V);
}

int trova(lab V[], int N, char so[]) {
  int i;

  for(i=0; i<N; i++)
    if(!strcmp(V[i].so, so) && !V[i].assegnata) return i;
  return -1;
} /* fine domanda 3 */
```

In un parcheggio a pagamento, la tariffa è proporzionale al numero di ore di permanenza di ogni veicolo all'interno del parcheggio. Ogni cliente, quindi, paga all'uscita, una volta calcolato per quante ore è stato utilizzato il parcheggio. A questo scopo, si realizzi un programma C che faccia uso del file di dati **parcheggiate.dat**. In particolare, **parcheggiate.dat** rappresenta l'insieme delle macchine presenti inizialmente all'interno del parcheggio; pertanto ogni record del file ha una struttura del tipo:

- **targa**: un codice alfanumerico che rappresenta la targa della macchina;
- **data**: un intero appartenente all'intervallo [1 … 366] che rappresenta la data in cui la macchina è stata collocata all'interno del parcheggio;
- **ora**: l'ora in cui la macchina è stata collocata all'interno del parcheggio.

Inoltre, il file contiene, all'inizio un valore intero che rappresenta il numero di macchine presenti al suo interno.

Il parcheggio prevede inoltre due tipi di tariffe: la tariffa oraria **A** relativa a clienti **abbonati** e la tariffa oraria **B** per clienti **non abbonati**. A questo proposito si prevede un secondo file binario **clienti.dat** che contiene le informazioni relative ai clienti abbonati:

- **cognome**: una stringa che contiene il nome del cliente;
- **targa**: un codice alfanumerico che rappresenta la targa della macchina posseduta dal cliente.

Il programma C dovrà:

1. A partire dal file **parcheggiate.dat**, creare un vettore **V** di dimensione opportuna contenente le informazioni sulle macchine parcheggiate.
2. A partire dal vettore V e dal file **clienti.dat** creare due vettori **W1** e **W2** che contengano, rispettivamente, l'insieme delle macchine di abbonati (W1) e di non abbonati (W2) presenti all'interno del parcheggio. A tal scopo si supponga l'esistenza di una funzione `abbonato` che, data una stringa rappresentante una targa, restituisca 0 se e solo se tale targa appartiene ad un cliente non abbonato (ovvero non presente nel file **clienti.dat**). Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima.
3. Dati da tastiera i dati (**targa**, **data** ed **ora** di uscita) di alcune macchine in uscita, stampare su schermo, per ogni macchina specificata, il costo del biglietto (= numero di ore * tariffa). Per il confronto tra stringhe si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche.
4. Si scriva il codice della funzione `abbonato`. Per il confronto tra stringhe si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define A 100
#define B 150

typedef struct { char targa[10]; int data, ora; } auto;
typedef struct { char cognome[30], targa[10]; } cliente;

main() {
  FILE *fp=fopen("parcheggiate.dat", "rb");
  int N, N1=0, N2=0, i, trovato;
  auto *V, *W1, *W2;
  cliente c;
  fread(&N, sizeof(int), 1, fp);
  V=(auto *)malloc(N*sizeof(auto));
  fread(V, sizeof(auto), N, fp);
  fclose(fp); /* fine domanda 1 */
  W1=(auto *)malloc(N*sizeof(auto));
  W2=(auto *)malloc(N*sizeof(auto));
  for(i=0; i<N; i++)
    if(abbonato(V[i].targa)) {
      strcpy(W1[N1].targa, V[i].targa);
      W1[N1].data=V[i].data;
      W1[N1++].ora=V[i].ora;
    }
    else {
      strcpy(W2[N2].targa, V[i].targa);
      W2[N2].data=V[i].data;
      W2[N2++].ora=V[i].ora;
    } /* fine domanda 2 */
  while(!feof(stdin)) {
    scanf("%s %d %d", c.targa, &c.data, &c.ora);
    trovato=0;
    for(i=0; (i<N1) && !trovato; i++)
      if(!strcmp(c.targa, W1[i].targa)) {
        printf("%d\n", A*((c.data-W1[i].data)*24+(c.ora-W1[i].ora)));
        trovato=1;
      }
    for(i=0; (i<N2) && !trovato; i++)
      if(!strcmp(c.targa, W2[i].targa)) {
        printf("%d\n", B*((c.data-W2[i].data)*24+(c.ora-W2[i].ora)));
        trovato=1;
      }
  } /* fine domanda 3 */
  free(W1); free(W2); free(V);
}

int abbonato(char s[]) {
  FILE *fp=fopen("clienti.dat", "rb");
  cliente c;
  while(!feof(fp)) {
    fread(&c, sizeof(cliente), 1, fp);
    if(!strcmp(s, c.targa)) {
      fclose(fp);
      return 1;
    }
  }
  fclose(fp);
  return 0;
} /* fine domanda 4 */
```

Per gestire il pagamento di ordini e fatture, una ditta utilizza due file:

- **ORDINI.DAT**, un file binario che contiene i dati relativi agli ordini effettuati. Ogni ordine è individuato da un numero identificativo univoco, il codice del fornitore, il codice dell'oggetto ordinato e la quantità di oggetti ordinati.
- **FATTURE.DAT**, un file binario che riporta i dati delle fatture ricevute dai fornitori. Per ciascuna fattura è specificato il numero identificativo dell'ordine, il codice del fornitore che l'ha emessa, il codice dell'oggetto ordinato, la quantità di oggetti forniti e l'importo totale.

Si realizzi un programma in linguaggio C che:

1. A partire dal file ORDINI.DAT, crei inizialmente un vettore V di dimensioni opportune (a tal scopo si supponga di richiedere all'utente il numero di elementi contenuti in ORDINI.DAT). In ciascun elemento del vettore V vengono inserite tutte le informazioni di ciascun record del file ORDINI.DAT e inoltre un campo booleano ***pagato*** (*vero* o *falso*) che indica se l'ordine è stato soddisfatto (inizialmente tale campo ha valore *falso* per ciascun elemento).
2. A partire dal file FATTURE.DAT e dal vettore V, scriva su un nuovo file binario (DA_PAGARE.DAT) le informazioni (codice del fornitore ed importo totale) relative alle fatture da pagare corrispondenti a ordini effettuati (ossia a elementi del vettore V) **completamente evasi** (cioè aventi quantità ordinata uguale a quantità fornita). Per ciascuno degli ordini evasi, si modifichi inoltre il valore del campo ***pagato*** nel vettore V ponendolo uguale a *vero*. A tal scopo si supponga l'esistenza di una funzione `trova_ord` che, dato un vettore di ordini, il codice di un ordine, il codice del fornitore ed il codice dell'oggetto ordinato, restituisca l'indice dell'ordine corrispondente all'interno del vettore.
3. Si scriva il codice della funzione `trova_ord`.

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct {
  int ident, forn, oggetto, quant;
} emissione;

typedef struct {
  emissione ord;
  int pagato;
} ordine;

int trova_ord(ordine *V, int ident, forn, oggetto);

main() {
  FILE *fp=fopen("ORDINI.DAT", "rb"), *fp1;
  int N, i=0, importo;
  emissione o;
  ordine *V;

  scanf("%d", &N);
  V=(ordine *)malloc(N*sizeof(ordine));
  while(!feof(fp)) {
    fread(&o, sizeof(emissione), 1, fp);
    V[i].ord.ident=o.ident;
    V[i].ord.forn=o.forn;
    V[i].ord.oggetto=o.oggetto;
    V[i].ord.quant=o.quant;
    V[i++].pagato=0;
  }
  fclose(fp); /* fine domanda 1 */
  fp=fopen("FATTURE.DAT", "rb");
  fp1=fopen("DA_PAGARE.DAT", "wb");
  while(!feof(fp)) {
    fread(&o, sizeof(emissione), 1, fp);
    fread(&importo, sizeof(int), 1, fp);
    i=trova_ord(V, o.ident, o.forn, o.oggetto);
    V[i].quant-=o.quant;
    if(V[i].quant==0) {
      V[i].pagato=1;
      fwrite(&(V[i].forn), sizeof(int), 1, fp);
      fwrite(&importo, sizeof(int), 1, fp);
    }
  }
  fclose(fp);
  fclose(fp1); /* fine domanda 2 */
  free(V);
}

int trova_ord(ordine *V, int ident, forn, oggetto) {
  int i;

  for(i=0; ; i++)
    if((V[i].ident==ident) && (V[i].forn==forn) && (V[i].oggetto==oggetto))
      return i;
} /* fine domanda 3 */
```

Ogni commissione degli esami di laurea, nella Facoltà di Ingegneria dell'Università di Bologna, è costituita da 10 docenti componenti, dei quali uno (necessariamente un professore ordinario) riveste il ruolo di Presidente. Gli altri nove componenti vengono scelti nelle tre fasce degli Ordinari, Associati e Ricercatori. Per gestire le commissioni di laurea, la segreteria di presidenza della facoltà utilizza due file:

- **DOCENTI.DAT**, un file binario che contiene i dati relativi al corpo docente. In particolare, per ogni persona, tale file riporta il nome, il codice (numero intero positivo univoco) e il ruolo (ricercatore, associato, ordinario).
- **COMMISSIONI.DAT**, un file binario che riporta i dati delle commissioni di laurea. Per ciascuna commissione è specificato il codice del presidente (che è un ordinario) ed i codici dei restanti nove commissari.

Si realizzi un programma C che:

1. A partire dal file DOCENTI.DAT, crei un vettore V di dimensione opportuna contenente le informazioni sui docenti (a tal fine, si supponga che il numero di elementi contenuti nel file venga richiesto all'utente da tastiera prima della lettura del file stesso).
2. A partire dal vettore V, crei due vettori ORDINARI ed ALTRI che contengano le informazioni (**nome** e **codice**) relative rispettivamente agli ORDINARI ed agli altri docenti (ASSOCIATI e RICERCATORI). Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima.
3. A partire dal vettore ORDINARI e dal file COMMISSIONI.DAT, visualizzi su schermo il nome ed il codice del primo docente che non è mai stato presidente di una commissione.
4. Letti a terminale altri nove codici per gli altri nove componenti della commissione, verifichi nei 2 vettori ORDINARI ed ALTRI, se tali codici esistono. In caso affermativo (cioè se tutti i codici esistono) si aggiunga al file COMMISSIONI.DAT la nuova commissione così composta (il presidente determinato al punto 3. + i nove componenti appena verificati).

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct { char nome[50]; int codice, ruolo; } docente;
typedef struct { int pres, comm[9]; } commissione;

main() {
  FILE *fp=fopen("DOCENTI.DAT", "rb");
  int N, i, N_ORD=0, N_ALT=0, pres, trovato=1, ok, j;
  docente *V, *ORDINARI, *ALTRI;
  commissione c;
  scanf("%d", &N);
  V=(docente *)malloc(N*sizeof(docente));
  fread(V, sizeof(docente), N, fp);
  fclose(fp); /* fine domanda 1 */
  ORDINARI=(docente *)malloc(N*sizeof(docente));
  ALTRI=(docente *)malloc(N*sizeof(docente));
  for(i=0; i<N; i++)
    if(V[i].ruolo==0) { /* docente ordinario */
      strcpy(ORDINARI[N_ORD].nome, V[i].nome);
      ORDINARI[N_ORD].codice=V[i].codice;
      ORDINARI[N_ORD ++].ruolo=0;
    }
    else {
      strcpy(ALTRI[N_ALT].nome, V[i].nome);
      ALTRI[N_ALT].codice=V[i].codice;
      ALTRI[N_ALT++].ruolo=V[i].ruolo;
    } /* fine domanda 2 */
  for(pres=0; trovato; pres++) {
    trovato=0;
    fp=fopen("COMMISSIONI.DAT", "rb");
    while(!feof(fp)&&!trovato) {
      fread(&c, sizeof(commissione), 1, fp);
      if(c.pres==ORDINARI[pres]) trovato=1;
    }
    fclose(fp);
  }
  printf("%d\n", pres); /* fine domanda 3 */
  c.pres=pres;
  for(i=0; i<9; i++) {
    scanf("%d", &(c.comm[i]));
    ok=0;
    for(j=0; (j<N_ORD)&&!ok; j++) if(ORDINARI[j].codice==c.comm[i]) ok=1;
    for(j=0; (j<N_ALT)&&!ok; j++) if(ALTRI[j].codice==c.comm[i]) ok=1;
  }
  if(ok) {
    fp=fopen("COMMISSIONI.DAT", "ab");
    fwrite(&c, sizeof(commissione), 1, fp);
    fclose(fp);
  } /* fine domanda 4 */
  free(V);
  free(ORDINARI);
  free(ALTRI);
}
```

Per gestire i prestiti di libri, una biblioteca di dipartimento utilizza due file:

- **LIBRI.DAT**, un file binario che contiene i dati relativi ai libri. Per ogni libro, tale file riporta il titolo, il numero di inventario (intero positivo univoco) ed un campo intero (*disponibile*) che indica se il libro è presente in biblioteca (disponibile = 1) o è in prestito (disponibile = 0). Il numero di libri presenti nella biblioteca è presente come numero intero all'inizio del file.
- **UTENTI.DAT**, un file binario che riporta i dati delle persone che possono avere libri in prestito. Per ciascuna persona è specificato il nome, l'indirizzo ed il numero di inventario del libro attualmente in prestito (al massimo uno). Il valore zero, come numero di inventario, indica nessun libro.

Si realizzi un programma in linguaggio C che:

1. A partire dal file LIBRI.DAT, crei un vettore V di dimensioni opportune contenente le informazioni sui libri.
2. A partire dal vettore V, crei due vettori PRESENTI ed IN_PRESTITO. Si inseriscano i dati dei libri presenti in biblioteca nella prima lista (PRESENTI) e quelli degli altri libri nella seconda lista (IN_PRESTITO). Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima.
3. Dato da tastiera il numero di inventario di un libro, acceda al vettore IN_PRESTITO per determinare se tale libro è in prestito e, in caso affermativo, accedendo al file UTENTI.DAT, stampi a video il nome della persona che lo ha in prestito.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  char titolo[50];
  int codice, disponibile;
} libro;

typedef struct {
  char nome[50], indirizzo[50];
  int codice;
} cliente;

main() {
  FILE *fp=fopen("LIBRI.DAT", "rb");
  int N, N_PR=0, N_PREST=0, libro;
  libro *V, *PRESENTI, *IN_PRESTITO;
  cliente c;

  fread(&N, sizeof(int), 1, fp);
  V=(libro *)malloc(N*sizeof(libro));
  fread(V, sizeof(libro), N, fp);
  fclose(fp); /* fine domanda 1 */
  PRESENTI=(libro *)malloc(N*sizeof(libro));
  IN_PRESTITO=(libro *)malloc(N*sizeof(libro));
  for(i=0; i<N; i++)
    if(V[i].disponibile) {
      strcpy(PRESENTI[N_PR].titolo, V[i].titolo);
      PRESENTI[N_PR].codice=V[i].codice;
      PRESENTI[N_PR++].disponibile=1;
    }
    else {
      strcpy(IN_PRESTITO[N_PREST].titolo, V[i].titolo);
      IN_PRESTITO[N_PREST].codice=V[i].codice;
      IN_PRESTITO[N_PREST ++].disponibile=0;
    } /* fine domanda 2 */
  scanf("%d", &libro);
  for(i=0; i<N_PREST, i++)
    if(libro==IN_PRESTITO[i]) {
      fp=fopen("UTENTI.DAT", "rb");
      do fread(&c, sizeof(cliente), 1, fp);
      while(c.codice!=libro);
      fclose(fp);
      printf("%s\n", c.nome);
      break;
    } /* fine domanda 3 */
  free(PRESENTI);
  free(IN_PRESTITO);
  free(V);
}
```

Per gestire gli iscritti ad uno Sci Club vengono utilizzati due file:

- **ASSOCIATI.DAT**, un file binario che riporta i dati delle persone associate al gruppo. Per ciascuna persona è specificato il nome, l'indirizzo ed un intero che rappresenta il tipo di attività praticata (il valore 1 sta per "Sci da discesa", 2 sta per "Sci da Fondo", 3 sta per "Snow Board"); a questo proposito si supponga che ogni associato possa specificare **una sola** attività praticata. Il numero totale di membri del club viene riportato all'inizio del file come un numero intero.
- **GITE.DAT**, un file binario che contiene i dati relativi alle prossime escursioni. Per ciascuna escursione, tale file riporta il nome della località, la data della gita ed il tipo di attività prevista.

Si realizzi un programma in linguaggio C che:

1. A partire dal file ASSOCIATI.DAT, crei un vettore V di dimensioni opportune contenente le informazioni sui soci del club.
2. Dato il nome di un associato letto da tastiera, acceda al file GITE.DAT ed al vettore V, stampi su schermo le informazioni (data e località) relative alle gite alle quali tale associato può partecipare (cioè quelle in cui l'attività prevista è la stessa praticata dalla persona). Per il confronto tra stringhe si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  char nome[50], indirizzo[50];
  int sport;
} socio;

typedef struct {
  char localita[30];
  int giorno, mese, anno, sport;
} gita;

main() {
  FILE *fp=fopen("ASSOCIATI.DAT", "rb");
  int N;
  char nome[50];
  socio *V;
  gita g;

  fread(&N, sizeof(int), 1, fp);
  V=(socio *)malloc(N*sizeof(socio));
  fread(V, sizeof(socio), N, fp);
  fclose(fp); /* fine domanda 1 */
  scanf("%s", nome);
  for(i=0; strcmp(nome, V[i].nome); i++);
  fp=fopen("GITE.DAT", "rb");
  while(!feof(fp)) {
    fread(&g, sizeof(gita), 1, fp);
    if(g.sport==V[i].sport) printf("%d %d %d %s\n", giorno, mese, anno, localita);
  }
  fclose(fp); /* fine domanda 2 */
  free(V);
}
```

Un concorso pubblico consiste di due prove: una prova **scritta** ed una prova **orale**. Ogni candidato al concorso, per poter essere inserito nella graduatoria finale, deve sostenere entrambe le prove. È possibile, comunque, che uno o più candidati ne sostengano una sola (lo scritto oppure l'orale): in tal caso i candidati vengono automaticamente esclusi dalla graduatoria finale. L'esito di ogni prova è rappresentato da un punteggio intero compreso tra 0 e 60 ed il punteggio finale di ciascun candidato è dato dalla somma dei punteggi conseguiti nelle due prove.

Si realizzi un programma C per gestire i risultati relativi al concorso. A questo scopo è dato un file binario "RISULTATI.DAT" che contiene l'elenco dei risultati ottenuti nelle due prove dai candidati, memorizzati in ordine casuale. Il particolare, ogni record di "RISULTATI.DAT" rappresenta l'esito di una prova (scritta o orale) di un certo candidato, e contiene quindi le seguenti informazioni:

- **nome**: una stringa che contiene il nome del candidato;
- **cognome**: una stringa che contiene il cognome del candidato;
- **prova**: un carattere che indica il tipo di prova ('s' significa scritto, 'o' indica orale);
- **punteggio**: un intero (compreso tra 0 e 60) che rappresenta il punteggio ottenuto dal candidato nella prova considerata.

All'inizio del file è contenuto un valore intero che rappresenta il numero di record contenuti nel file.

Il programma dovrà realizzare i seguenti punti:

1. A partire dal file RISULTATI.DAT costruisca un vettore V di dimensione opportuna contenente le informazioni sui risultati di entrambe le prove.
2. A partire da V, crei due vettori WS e WO che contengano i risultati (nome, cognome e punteggio) conseguiti, rispettivamente, nella prova scritta (WS) e nella prova orale (WO) dai vari candidati. Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima.
3. A partire dai due vettori WS e WO costruite al punto precedente, si costruisca un terzo vettore WG che rappresenti la graduatoria finale del concorso. Ogni elemento di WG ha la stessa struttura degli elementi di WS e WO (nome, cognome e punteggio) e rappresenta il **risultato finale** ottenuto da un particolare candidato. In WG deve essere quindi inserito un elemento per ogni candidato che abbia sostenuto entrambe le prove (ovvero che abbia un risultato sia in WS che in WO). In particolare, il campo punteggio di ogni elemento di WG conterrà la somma dei due punteggi conseguiti nelle due prove dal candidato. Per il confronto tra stringhe si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche.
4. Si stampi a video il contenuto del vettore WG.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  char nome[20], cognome[30];
  int punteggio;
} esito;

typedef struct {
  esito e;
  char prova;
} risultato;

main() {
  FILE *fp=fopen("RISULTATI.DAT", "rb");
  int N, NS=0, NO=0, NG=0, i, j;
  risultato *V;
  esito *WS, *WO, *WG;

  fread(&N, sizeof(int), 1, fp);
  V=(risultato *)malloc(N*sizeof(risultato));
  fread(V, sizeof(risultato), N, fp);
  fclose(fp); /* fine domanda 1 */
  WS=(esito *)malloc(N*sizeof(esito));
  WO=(esito *)malloc(N*sizeof(esito));
  for(i=0; i<N; i++)
    if(V[i].prova='s') {
      strcpy(WS[NS].nome, V[i].e.nome);
      strcpy(WS[NS].cognome, V[i].e.cognome);
      WS[NS++].punteggio=V[i].e.punteggio;
    }
    else {
      strcpy(WO[NO].nome, V[i].e.nome);
      strcpy(WO[NO].cognome, V[i].e.cognome);
      WO[NO++].punteggio=V[i].e.punteggio;
    } /* fine domanda 2 */
  WG=(esito *)malloc(N*sizeof(esito));
  for(i=0; i<NS; i++)
    for(j=0; j<NO; j++)
      if(!strcmp(WS[i].nome, WO[j].nome)&&! strcmp(WS[i].cognome, WO[j].cognome)) {
        strcpy(WG[NG].nome, WS[i].nome);
        strcpy(WG[NG].cognome, WS[i].cognome);
        WG[NG++].punteggio=WS[i].punteggio+WO[j].punteggio;
      } /* fine domanda 3 */
  for(i=0; i<NG; i++)
    printf("%s %s %d\n", WG[i].nome, WG[i].cognome, WG[i].punteggio);
  /* fine domanda 4 */
  free(WG);
  free(WS);
  free(WO);
  free(V);
}
```

Un grande magazzino vende merci appartenenti a 2 categorie: alimentari e casalinghi. Tutti gli articoli trattati dal grande magazzino sono registrati in un file binario "**catalogo.dat**" contenente le informazioni relative a ciascun articolo trattato. In particolare, ogni record di "catalogo.dat" ha la seguente struttura:

- **codice**: un valore intero che individua univocamente l'articolo;
- **descrizione**: una stringa di 40 caratteri contenente una descrizione testuale dell'articolo;
- **fornitore**: una stringa di 30 caratteri contenente il nome dell'azienda che ha fornito l'articolo al grande magazzino;
- **prezzo**: un intero che contiene il prezzo al pubblico dell'articolo;
- **costo**: un intero che rappresenta il prezzo di acquisto dell'articolo (applicato dal fornitore al grande magazzino).

Un secondo file di testo "**alimenti.txt**" contiene l'elenco degli articoli appartenenti alla categoria degli alimentari trattati dal magazzino: in particolare ogni riga del file contiene un intero che rappresenta il codice di un articolo presente nel file catalogo.dat.

Si realizzi un programma C che:

1. A partire dal file catalogo.dat, crei un vettore V di dimensione opportuna (a tal fine, si supponga che il numero di elementi contenuti nel file venga richiesto all'utente tramite tastiera prima della lettura del file stesso).
2. A partire dal vettore V e dal file alimenti.txt realizzi **due vettori A e C** contenenti rispettivamente le informazioni (codice, descrizione, fornitore, prezzo, costo) relative agli alimentari (vettore A) ed ai casalinghi (vettore C). Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima.
3. Letto da tastiera un valore intero positivo S, a partire da A e C costruisca un quarto vettore L contenente gli articoli per i quali il **guadagno percentuale** (vedi definizione in seguito) è minore o uguale al valore S.

Definizione di "guadagno percentuale": dato un articolo A, sia P il suo prezzo e C il suo costo; il guadagno percentuale G associato ad A è definito dalla formula:

$$G = ((P - C) / C) * 100$$

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  int codice, prezzo, costo;
  char descr[40], forn[30];
} articolo;

main() {
  FILE *fp=fopen("catalogo.dat", "rb");
  int N, i, NA=0, NC=0, trovato, codice, S, NL=0;
  articolo *V, *A, *C, *L;

  scanf("%d", &N);
  V=(articolo *)malloc(N*sizeof(articolo));
  fread(V, sizeof(articolo), N, fp);
  fclose(fp); /* fine domanda 1 */
  A=(articolo *)malloc(N*sizeof(articolo));
  C=(articolo *)malloc(N*sizeof(articolo));
  for(i=0; i<N; i++) {
    trovato=0;
    fp=fopen("alimenti.txt", "rt");
    while(!feof(fp))&&!trovato) {
      fscanf(fp, "%d", &codice);
      if(codice==V[i].codice) {
        A[NA].codice=codice;
        A[NA].prezzo=V[i].prezzo;
        A[NA].costo=V[i].costo;
        strcpy(A[NA].descr, V[i].descr);
        strcpy(A[NA++].forn, V[i].forn);
        trovato=1;
      }
    }
    fclose(fp);
    if(!trovato) {
      C[NC].codice=codice;
      C[NC].prezzo=V[i].prezzo;
      C[NC].costo=V[i].costo;
      strcpy(C[NC].descr, V[i].descr);
      strcpy(C[NC++].forn, V[i].forn);
    }
  } /* fine domanda 2 */
  scanf("%d", &S);
  L=(articolo *)malloc(N*sizeof(articolo));
  for(i=0; i<NA; i++)
    if((A[i].prezzo-A[i].costo)/A[i].costo*100<=S) {
      L[NL].codice=A[i].codice;
      L[NL].prezzo=A[i].prezzo;
      L[NL].costo=A[i].costo;
      strcpy(L[NL].descr, A[i].descr);
      strcpy(L[NL++].forn, A[i].forn);
    }
  for(i=0; i<NC; i++) {
    if((C[i].prezzo-C[i].costo)/C[i].costo*100<=S) {
      L[NL].codice=C[i].codice;
      L[NL].prezzo=C[i].prezzo;
      L[NL].costo=C[i].costo;
      strcpy(L[NL].descr, C[i].descr);
      strcpy(L[NL++].forn, C[i].forn);
    } /* fine domanda 3 */
  free(L);
  free(A);
  free(C);
  free(V);
}
```

Una casa editrice pubblica due riviste R1 e R2. I due periodici sono venduti (oltre che in edicola) anche in abbonamento postale: siano P1 e P2 i prezzi di abbonamento rispettivamente relativi a R1 e ad R2. Per gestire gli abbonamenti postali alle riviste, sono dati tre file:

- "**clienti.dat**" è un file binario che contiene le informazioni relative ai clienti che hanno sottoscritto uno (o eventualmente due) abbonamenti; in particolare, ciascun record di clienti.dat contiene:
  - **codice**: un intero che identifica univocamente il cliente;
  - **nome**, **cognome**: rappresentati da stringhe;
  - **indirizzo**: una stringa che contiene l'indirizzo del cliente.
- "**abbonamenti_R1.txt**" è un file di **testo** che contiene, per ogni riga, i codici dei clienti abbonati alla rivista R1;
- "**abbonamenti_R2.txt**" è un file di **testo** che contiene, per ogni riga, i codici dei clienti abbonati alla rivista R2;

Si realizzi un programma C che realizzi i seguenti punti:

1. A partire dal file "**clienti.dat**" crei un vettore V di dimensioni opportune contenente le informazioni sui clienti (a tal fine, si supponga che il numero di elementi contenuti nel file venga richiesto all'utente via input prima della lettura del file stesso).
2. A partire dal vettore V e dai file "**abbonamenti_R1.txt"** e "**abbonamenti_R2.txt**.", crei due vettori W1 e W2 contenenti rispettivamente le informazioni (**codice**, **nome**, **cognome**, **indirizzo**) relative agli abbonati rispettivamente della rivista R1 (vettore W1) e della rivista R2 (vettore W2). A tal scopo si supponga l'esistenza di una funzione `trova_cli` che, dato un vettore di clienti e il codice di un cliente, restituisca l'indice corrispondente al cliente considerato. Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima.
3. A partire dai due vettori W1 e W2, si costruisca un file di testo "**VENDITE_R1.TXT**" in cui, per ogni abbonato della rivista R1, sia riportato nome, cognome e prezzo di abbonamento (ad R1). Il prezzo può variare a causa delle politiche promozionali della casa editrice: se un abbonato di R1 è abbonato anche ad R2, usufruirà di uno sconto del 25% sul prezzo di abbonamento ad R1.
4. Si scriva il codice della funzione `trova_cli`.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int P1=100, p2=150;

typedef struct {
  int codice;
  char nome[20], cognome[30], indirizzo[50];
} cliente;

int trova_cli(cliente V[], int codice);

main() {
  FILE *fp=fopen("clienti.dat", "rb");
  int N, i, N1=0, N2=0, cli;
  cliente *V, *W1, *W2;

  scanf("%d", &N);
  V=(cliente *)malloc(N*sizeof(cliente));
  fread(V, sizeof(cliente), N, fp);
  fclose(fp); /* fine domanda 1 */
  W1=(cliente *)malloc(N*sizeof(cliente));
  W2=(cliente *)malloc(N*sizeof(cliente));
  fp=fopen("abbonamenti_R1.txt", "rt");
  while(!feof(fp)) {
    fscanf(fp, "%d", &cli);
    i=trova_cli(V, cli);
    W1[N1].codice=cli;
    strcpy(W1[N1].nome, V[i].nome);
    strcpy(W1[N1].cognome, V[i].cognome);
    strcpy(W1[N1++].indirizzo, V[i].indirizzo);
  }
  fclose(fp);
  fp=fopen("abbonamenti_R2.txt", "rt");
  while(!feof(fp)) {
    fscanf(fp, "%d", &cli);
    i=trova_cli(V, cli);
    W2[N2].codice=cli;
    strcpy(W2[N2].nome, V[i].nome);
    strcpy(W2[N2].cognome, V[i].cognome);
    strcpy(W2[N2++].indirizzo, V[i].indirizzo);
  }
  fclose(fp); /* fine domanda 2 */
  fp=fopen("VENDITE_R1.TXT", "wt");
  for(i=0; i<N1; i++) {
    int j, trovato=0;

    for(j=0; (j<N2) && !trovato; j++)
      if(W1[i].codice==W2[j].codice) {
        fprintf(fp, "%s %s %d\n", W1[i].nome, W1[i].cognome, P1*0.75);
        trovato=1;
      }
    if(!trovato) fprintf(fp, "%s %s %d\n", W1[i].nome, W1[i].cognome, P1);
  }
  fclose(fp); /* fine domanda 3 */
  free(W1);
  free(W2);
  free(V);
}
```

```c
int trova_cli(cliente V[], int codice) {
  int i=0;

  while(V[i].codice!=codice) i++;
  return i;
} /* fine domanda 4 */
```

Un laboratorio di analisi mediche opera all'interno di un ospedale. In particolare, il laboratorio effettua analisi di vario tipo sia per pazienti **interni** all'ospedale (cioè ricoverati) che per pazienti **esterni** (non ricoverati). Per gestire le informazioni relative alle analisi da effettuare, il laboratorio utilizza due file binari: "analisi.dat" e "ricoverati.dat".

Il file "**analisi.dat**" contiene le informazioni relative alle analisi in corso presso il laboratorio; in particolare, ogni record del file "analisi.dat" rappresenta una analisi e ha la seguente struttura:

- **numero**: un intero che individua univocamente l'analisi (non possono cioè esistere più record con lo stesso numero all'interno del file);
- **codice**: un intero che individua univocamente il paziente (sia interno che esterno);
- **tipo_analisi**: una stringa che contiene una sequenza di caratteri che individua univocamente il tipo di analisi richiesta;
- **data_referto**: un intero (da 1 a 366) che indica il giorno a partire dal quale possono essere ritirati i risultati dell'analisi.

Il file "**ricoverati.dat**" contiene le informazioni relative ai pazienti ricoverati presso l'ospedale; in particolare, ogni record del file "ricoverati.dat" rappresenta un ricoverato e ha la seguente struttura:

- **codice**: un intero che individua univocamente il paziente;
- **codice reparto**: una stringa che contiene una sequenza di caratteri che indica univocamente il reparto presso il quale il paziente è ricoverato;

Si realizzi un programma C che risolva i seguenti punti:

1. A partire dal file "analisi.dat" si crei un vettore V di dimensioni opportune contenente le informazioni sulle analisi (a tal fine, si supponga che il numero di elementi contenuti nel file venga richiesto all'utente via input prima della lettura del file stesso).
2. A partire dal vettore V e dal file "ricoverati.dat" si costruiscano **due vettori W1 e W2** contenenti rispettivamente le informazioni (numero, codice, tipo_analisi, data_referto) relative alle analisi di pazienti **interni** (W1) e di pazienti **esterni** (W2). Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima.
3. Dato in ingresso **un intero G** (compreso tra 1 e 366), il programma stampi sullo schermo **numero** e **codice** delle analisi (richieste sia da pazienti interni che esterni) per le quali il referto **è già disponibile il giorno G** (cioè quelle con **data_referto <= G**). I dati (**numero**, **codice**, **tipo_analisi**, **data_referto**) relativi alle analisi il cui referto non è ancora pronto il giorno G devono essere invece riscritti nel file binario "**analisi.dat**".

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  int numero, codice, data;
  char tipo[30];
} analisi;

typedef struct {
  int codice, reparto;
} paziente;

main() {
  FILE *fp=fopen("analisi.dat", "rb");
  int N, i, N1=0, N2=0, G;
  analisi *V, *W1, *W2;
  paziente p;

  scanf("%d", &N);
  V=(analisi *)malloc(N*sizeof(analisi));
  fread(V, sizeof(analisi), N, fp);
  fclose(fp); /* fine domanda 1 */
  W1=(analisi *)malloc(N*sizeof(analisi));
  W2=(analisi *)malloc(N*sizeof(analisi));
  for(i=0; i<N; i++) {
    int trovato=0;

    fp=fopen("analisi.dat", "rb");
    while(!feof(fp)&&!trovato) {
      fread(&p, sizeof(paziente), 1, fp);
      if(V[i].codice==p.codice) {
        W1[N1].numero=V[i].numero;
        W1[N1].codice=V[i].codice;
        W1[N1].data=V[i].data;
        strcpy(W1[N1++].tipo, V[i].tipo);
        trovato=1;
      }
    }
    if(!trovato) {
      W2[N2].numero=V[i].numero;
      W2[N2].codice=V[i].codice;
      W2[N2].data=V[i].data;
      strcpy(W2[N2++].tipo, V[i].tipo);
    }
    fclose(fp);
  } /* fine domanda 2 */
  scanf("%d", &G);
  fp=fopen("analisi.dat", "wb");
  for(i=0; i<N; i++)
    if(V[i].data<=G) printf("%d %d", V[i].numero, V[i].codice);
    else fwrite(&(V[i]), sizeof(analisi), 1, fp);
  fclose(fp); /* fine domanda 3 */
  free(W1);
  free(W2);
  free(V);
}
```

Un’azienda produttrice di articoli di abbigliamento vende i propri capi attraverso due negozi N1 e N2. In particolare, per gestire le vendite, ogni negozio mantiene le informazioni in un file: siano **vendite1.dat** e **vendite2.dat** i file binari associati rispettivamente al negozio N1 ed al negozio N2; in particolare, ogni record di vendite1.dat e vendite2.dat rappresenta un capo acquistato ed ha la seguente struttura:

- **codice**: una stringa di 5 caratteri che identifica **univocamente** l’articolo venduto;
- **descrizione**: una stringa di 25 caratteri che contiene la descrizione dell’articolo;
- **taglia**: un intero che indica la taglia dell’articolo acquistato;
- **colore**: un intero che individua univocamente il colore del capo acquistato;
- **prezzo**: il prezzo di vendita del capo.

All’inizio di ogni file è contenuto un valore intero che indica il numero totale di record contenuti nel file stesso.

Si realizzi un programma C che risolva i seguenti punti:

1. A partire dai file vendite1.dat e vendite2.dat, crei un vettore V di dimensioni opportune in cui sono registrate tutte le vendite effettuate presso i due negozi.
2. Stampi l’incasso totale (dato dalla somma di tutti i prezzi dei capi venduti).
3. A partire da V e sapendo che la gamma dei colori possibili è rappresentata da codici (interi) appartenenti all’intervallo [0 … MAX] (dove MAX è una costante data), stampi il codice del colore “**più venduto**” (cioè il codice del colore che è presente nel maggior numero di elementi del vettore V).

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 15

typedef struct {
  char codice[6], descrizione[26];
  int taglia, colore, prezzo;
} capo;

main() {
  FILE *fp1=fopen("vendite1.dat", "rb"), *fp2=fopen("vendite2.dat", "rb");
  int N1, N2, i, tot=0, max=0, imax;
  capo *V;
  int colori[MAX];

  fread(&N1, sizeof(int), 1, fp1);
  fread(&N2, sizeof(int), 1, fp2);
  V=(capo *)malloc((N1+N2)*sizeof(capo));
  fread(V, sizeof(capo), N1, fp1);
  fread(V+N1, sizeof(capo), N2, fp2);
  fclose(fp1);
  fclose(fp2); /* fine domanda 1 */
  for(i=0; i<(N1+N2); i++) tot+=V[i].prezzo;
  printf("%d\n", tot); /* fine domanda 2 */
  for(i=0; i<MAX; i++) colori[i]=0;
  for(i=0; i<(N1+N2); i++) colori[V[i].colore]++;
  for(i=0; i<MAX; i++)
    if(colori[i]>max) {
      max=colori[i];
      imax=i;
    }
  printf("%d\n", imax); /* fine domanda 3 */
  free(V);
}
```

Si devono organizzare i saldi di un grande magazzino. A questo scopo, siano dati 2 file binari "articoli.dat" e "saldi.dat". Il file "articoli.dat" rappresenta l'archivio di tutti gli articoli in vendita; in particolare, ogni record di tale file contiene le seguenti informazioni:

- **codice**: un intero che individua univocamente l'articolo;
- **descrizione**: una stringa che contiene la descrizione dell'articolo;
- **marca**: una stringa che rappresenta la casa produttrice dell'articolo;
- **codice_reparto**: un intero che individua univocamente il reparto in cui l'articolo è in vendita;
- **prezzo**: un intero che rappresenta il prezzo di vendita dell'articolo.

Il file "saldi.dat" contiene l'elenco degli articoli in saldo; in particolare, ciascun record di tale file contiene le seguenti informazioni:

- **codice**: un intero che individua univocamente l'articolo;
- **sconto**: un intero (compreso tra 0 e 100) che rappresenta lo sconto (in percentuale) da applicare al prezzo di vendita dell'articolo.

Entrambi i file contengono, come primo valore, un intero rappresentante il numero di record presenti all'interno del file stesso.

Si realizzi un programma C che, risolva i seguenti punti:

1. A partire dal file binario "articoli.dat", crei un vettore V di dimensioni opportune contenente le informazioni sugli articoli.
2. A partire dal vettore V, crei due vettori W1 e W2 contenenti rispettivamente le informazioni (codice, descrizione, marca, codice_reparto e prezzo) relative agli articoli non scontati (W1) ed agli articoli in saldo (W2). In particolare, gli elementi del vettore W2 (articoli scontati) conterranno nel campo "prezzo" il prezzo reale (ottenuto applicando al prezzo di vendita lo sconto indicato). A tal scopo si supponga l'esistenza di una funzione `saldi` che, dato il codice di un articolo, acceda al file "saldi.dat" e restituisca lo sconto applicato se l'articolo in questione è in saldo e 0 altrimenti. Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all'interno del file "string.h", che, date due stringhe, copia il contenuto della seconda nella prima.
3. Letto da tastiera un intero R, si scrivano in un terzo file di testo "sconti.txt" le informazioni (codice, descrizione, marca, codice_reparto e prezzo) relative agli articoli in saldo in vendita presso il reparto il cui codice è R.
4. Si scriva il codice della funzione `saldi`.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  int codice, reparto, prezzo;
  char descr[30], marca[20];
} articolo;

typedef struct { int codice, sconto; } saldo;

int saldi(int codice);

main() {
  FILE *fp=fopen("articoli.dat", "rb");
  int N, N1=0, N2=0, R;
  articolo *V, *W1, *W2;

  fread(&N, sizeof(int), 1, fp);
  V=(articolo *)malloc(N*sizeof(articolo));
  fread(V, sizeof(articolo), N, fp);
  fclose(fp); /* fine domanda 1 */
  W1=(articolo *)malloc(N*sizeof(articolo));
  W2=(articolo *)malloc(N*sizeof(articolo));
  for(i=0; i<N; i++) {
    int s= saldi(V[i].codice);

    if(s) {
      W1[N1].codice=V[i].codice;
      W1[N1].reparto=V[i].reparto;
      strcpy(W1[N1].descr, V[i].descr);
      strcpy(W1[N1].marca, V[i].marca);
      W1[N1++].prezzo=V[i].prezzo;
    }
    else {
      W2[N2].codice=V[i].codice;
      W2[N2].reparto=V[i].reparto;
      strcpy(W2[N2].descr, V[i].descr);
      strcpy(W2[N2].marca, V[i].marca);
      W2[N2++].prezzo=V[i].prezzo*(100-s)/100;
    } /* fine domanda 2 */
  scanf("%d", &R);
  fp=fopen("sconti.txt", "wt");
  for(i=0; i<N2; i++)
    if(W2[i].reparto==R)
      fprintf(fp, "%d %s %s %d %d\n", W2[i].codice, W2[i].descr, W2[i].marca, R,
        W2[i].prezzo);
  fclose(fp); /* fine domanda 3 */
  free(W1);
  free(W2);
  free(V);
}

int saldi(int codice) {
  FILE *fp=fopen("saldi.dat", "rb");
  saldo s;

  while(!feof(fp)) {
    fread(&s, sizeof(saldo), 1, fp);
    if(s.codice==codice) {
      fclose(fp);
      return s.sconto;
    }
  }
  fclose(fp);
  return 0;
} /* fine domanda 4 */
```

Si vuole automatizzare la gestione del magazzino di una concessionaria auto. A tale scopo, sia dato il file binario "archivio.dat" in cui sono memorizzati i dati relativi alle autovetture presenti nel magazzino della concessionaria. In particolare, ogni record di "archivio.dat" contiene le seguenti informazioni:

- **numero_telaio**: un intero che individua univocamente l'autovettura;
- **marca**: una stringa di 20 caratteri che rappresenta la casa produttrice dell'auto;
- **modello**: una stringa di 20 caratteri che rappresenta il modello dell'autovettura;
- **prezzo**: un intero che rappresenta il prezzo di vendita della macchina;
- **provenienza**: un carattere appartenente al dominio {'u', 'n'} che indica se l'autovettura è usata (valore 'u') oppure nuova (valore 'n');

Sia dato, inoltre, un secondo file binario "vendite.dat" contenente le informazioni relative alle auto già vendute presenti in magazzino. In particolare, ogni record di vendite.dat contiene i seguenti campi:

- **numero_telaio:** un intero che individua univocamente l'autovettura;
- **cliente**: una stringa di 30 caratteri che rappresenta il cognome dell'acquirente.

Si realizzi un programma C che realizzi i seguenti punti:

1. A partire dal file binario "archivio.dat" crei un vettore V di dimensioni opportune contenente le informazioni sulle auto (a tal fine, si supponga che il numero di elementi contenuti nel file venga richiesto all'utente da tastiera prima della lettura del file stesso).
2. A partire dal vettore V e dal file "vendite.dat" costruisca due vettori W1 e W2 contenenti le informazioni relative alle auto non ancora vendute (W1) ed alle auto già vendute (W2), rispettivamente.
3. Letta da tastiera una stringa M che rappresenta una marca, a partire dal vettore W1 delle auto disponibili, costruisca due vettori N e U:
   - N conterrà i dati delle auto **nuove** (cioè con Provenienza = 'n') e **disponibili** (cioè non ancora vendute) di marca M;
   - U conterrà i dati delle auto **usate** (cioè con Provenienza = 'u') e **disponibili** di marca M.

   Per il confronto tra stringhe si utilizzi la funzione `strcmp`, dichiarata all'interno del file "string.h", che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche.
4. Stampi il numero totale delle auto disponibili (sia nuove che usate) di marca M.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct {
  int telaio, prezzo;
  char marca[20], modello[20], provenienza;
} auto;
typedef struct {
  int telaio;
  char cliente[30];
} vendita;

main() {
  FILE *fp=fopen("archivio.dat", "rb");
  int N, N1=0, N2=0, NN=0, NU=0, i;
  auto *V, *W1, *W2, *N, *U;
  char M[20];
  scanf("%d", &N);
  V=(auto *)malloc(N*sizeof(auto));
  fread(V, sizeof(auto), N, fp);
  fclose(fp); /* fine domanda 1 */
  W1=(auto *)malloc(N*sizeof(auto));
  W2=(auto *)malloc(N*sizeof(auto));
  for(i=0; i<N; i++) {
    vendita v;
    int trovato=0;
    fp=fopen("vendite.dat", "rb");
    while(!feof(fp)&&!trovato) {
      fread(&v, sizeof(vendita), 1, fp);
      if(v.telaio=V[i].telaio) {
        W2[N2].telaio=V[i].telaio;
        W2[N2].prezzo=V[i].prezzo;
        strcpy(W2[N2].marca, V[i].marca);
        strcpy(W2[N2].modello, V[i].modello);
        W2[N2++].provenienza=V[i].provenienza;
        trovato=1;
      }
    }
    if(!trovato) {
      W1[N1].telaio=V[i].telaio;
      W1[N1].prezzo=V[i].prezzo;
      strcpy(W1[N1].marca, V[i].marca);
      strcpy(W1[N1].modello, V[i].modello);
      W1[N1++].provenienza=V[i].provenienza;
    }
    fclose(fp);
  } /* fine domanda 2 */
  scanf("%s", M);
  N=(auto *)malloc(N1*sizeof(auto));
  U=(auto *)malloc(N1*sizeof(auto));
  for(i=0; i<N1; i++) if(!strcmp(W1[i].marca, M))
       if(W1[i].provenienza=='n') {
         N[NN].telaio=V[i].telaio;
         N[NN].prezzo=V[i].prezzo;
         strcpy(N[NN].marca, M);
         strcpy(N[NN].modello, V[i].modello);
         N[NN++].provenienza='n';
       }
       else {
         U[NU].telaio=V[i].telaio;
         U[NU].prezzo=V[i].prezzo;
         strcpy(U[NU].marca, M);
         strcpy(U[NU].modello, V[i].modello);
         U[NU++].provenienza='u';
       } /* fine domanda 3 */
  printf("%d\n", NN+NU); /* fine domanda 4 */
  free(N); free(U); free(W1); free(W2); free(V);
}
```

Sia dato un file binario “archivio.dat” contenente i dati anagrafici dei pazienti ricoverati in un ospedale. Ogni record di “archivio.dat” contiene le informazioni relative ad una persona ricoverata nell’ospedale; in particolare:

- **codice**: un intero che indica univocamente il paziente;
- **nome:** una stringa che rappresenta il nome del paziente;
- **reparto:** una stringa che rappresenta il reparto in cui è ricoverato il paziente;
- **camera**: un intero che indica il numero della camera in cui è ricoverato il paziente;
- **patologia**: una stringa che indica la patologia per cui il paziente è stato ricoverato.

Il numero totale di pazienti viene indicato da un intero presente all’inizio del file.
Un secondo file binario “prestazioni.dat” contiene l’elenco delle prestazioni mediche erogate a pazienti ricoverati presso l’ospedale. In particolare, ogni record di “prestazioni.dat” contiene le seguenti informazioni:

- **codice**: il codice del paziente che ha effettuato il pagamento;
- **prestazione**: una stringa che indica la prestazione per alla quale si riferisce il pagamento;
- **ticket**: un intero che indica il costo (*ticket*) della prestazione.

Si realizzi un programma C che:

1. A partire dal file “archivio.dat”, crei un vettore V di dimensioni opportune che contenga i dati relativi ai pazienti ricoverati. Più precisamente, ogni elemento del vettore rappresenta un paziente e contiene: codice, nome, reparto, camera, totale (un intero che rappresenta l’ammontare totale di tutti i ticket relativi a prestazioni erogate al paziente). A tal scopo si supponga l’esistenza di una funzione `ticket` che, dato il codice di un paziente, acceda al file “prestazioni.dat” e restituisca l’ammontare totale dei ticket relativi a tale paziente. Per la copia di stringhe si utilizzi la funzione `strcpy`, dichiarata all’interno del file “string.h”, che, date due stringhe, copia il contenuto della seconda nella prima.
2. A partire dal vettore V e letta da tastiera una stringa S che rappresenta un reparto, stampi su schermo **nome**, **reparto**, e **camera** di tutti i pazienti ricoverati nel reparto S. Per il confronto tra stringhe si utilizzi la funzione `strcmp`, dichiarata all’interno del file “string.h”, che, date due stringhe, restituisce 0 se e solo se le due stringhe sono identiche.
3. Si scriva il codice della funzione `ticket`.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct {
  int codice, camera;
  char nome[50], reparto[30], patologia[50];
} paziente;

typedef struct {
  int codice, ticket;
  char prest[50];
} prestazione;

typedef struct {
  paziente p;
  int totale;
} ricoverato;

int ticket(int codice);

main() {
  FILE *fp=fopen("archivio.dat", "rb");
  int N, i=0;
  char S[30];
  ricoverato *V;

  fread(&N, sizeof(int), 1, fp);
  V=(ricoverato *)malloc(N*sizeof(ricoverato));
  while(!feof(fp)) {
    fread(&(V[i].paziente), sizeof(paziente), 1, fp);
    V[i++].totale=ticket(V[i].p.codice);
  }
  fclose(fp); /* fine domanda 1 */
  scanf("%s", S);
  for(i=0; i<N; i++)
    if(!strcmp(V[i].reparto, S)
      printf("%s %s %s\n, V[i].nome, S, V[i].camera); /* fine domanda 2 */
  free(V);
}
```

```c
int ticket(int codice) {
  FILE *fp=fopen("prestazioni.dat", "rb");
  int tot=0;
  prestazione p;

  while(!feof(fp)) {
    fread(&p, sizeof(prestazione), 1, fp);
    if(p.codice==codice)
      tot+=p.ticket;
  }
  fclose(fp);
  return tot;
} /* fine domanda 3 */
```