Istruzioni

- Tempo disponibile: 120 minuti.
- Non è permesso l'uso di dispositivi elettronici (a parte il PC della propria postazione).
- Il programma sarà valutato per
  - Identificazione delle strutture dati e degli algoritmi appropriati alle specifiche
  - Corretta implementazione di strutture dati e algoritmi
  - Utilizzo efficiente delle risorse
  - Stile (chiarezza, utilizzo di costrutti appropriati, corretta strutturazione)
- I programmi non compilabili saranno valutati 0 punti.
- Fare l'upload di tutti i file che compongono il programma.
- Visual Studio Code è installato su tutte le postazioni. Si può comunque usare l'editor che si preferisce fra quelli installati.

Esercizio - Parte 1 (max 17 punti)

Un file binario contiene i codici fiscali (array di 17 `char`, compreso il terminatore) degli iscritti a una palestra.

Codice fiscale
HVVFPL60A25L248H
CNZVVF49M48E182X
LPNVRR74D54A850W
ZVDVFZ62P51D595R

Tabella 1: Contenuto dell'allegato `codicifiscali.dat`

Ad esempio, il file allegato `codicifiscali.dat` contiene i dati mostrati in Tabella 1.

La palestra offre tre attività (indicate dagli interi `1`, `2` e `3`) e ogni iscrizione dà diritto a 4 accessi settimanali ad ogni attività.

Un file di testo, come l'allegato `ingressi.txt`, contiene i tentativi di ingresso di ogni utente ad una attività durante la settimana corrente. Ogni riga del file contiene il codice fiscale dell'utente e l'intero che identifica l'attività.

Si scriva un programma in linguaggio C, da compilare in un eseguibile di nome `palestra`, che verifichi che l'utente abbia diritto agli ingressi che tenta. In particolare, il programma deve

- prendere come argomenti della linea di comando i nomi, rispettivamente, di un file binario e di un file di testo dei formati sopra indicati;
- creare una lista collegata contenente un elemento per ogni utente (quindi, se il file binario è quello allegato, la lista dovrà avere 4 elementi). Si suggerisce che l'elemento della lista sia composto dal codice fiscale e da tre contatori degli accessi residui, uno per ogni attività, inizializzati a 4;
- per ogni tentativo di ingresso di un utente a un'attività, verificare che l'utente non abbia esaurito i suoi ingressi per quell'attività; in caso che gli ingressi siano esauriti, stampare un messaggio di accesso negato, altrimenti decrementare il contatore degli accessi dell'utente per l'attività. Ad esempio, a seguito del primo tentativo di ingresso contenuto nel file `ingressi.txt`, dovrebbe essere decrementato il contatore relativo all'attività 2 dell'utente di codice fiscale LPNVRR74D54A850W.

Inoltre, una volta elaborati tutti i tentativi di ingresso, il programma deve stampare l'elenco degli utenti, con il numero di ingressi residui per ogni attività.

Ad esempio, se `codicifiscali.dat` e `ingressi.txt` fossero i file allegati, l'invocazione `./palestra codicifiscali.dat ingressi.txt` dovrebbe produrre un output simile al seguente:

```
Codice Fiscale ZVDVFZ62P51D595R, attivita' 3: Accesso negato
Codice Fiscale LPNVRR74D54A850W, attivita' 3: Accesso negato
Codice Fiscale LPNVRR74D54A850W, attivita' 3: Accesso negato
Codice Fiscale ZVDVFZ62P51D595R, attivita' 3: Accesso negato
Codice Fiscale CNZVVF49M48E182X, attivita' 1: Accesso negato
Codice Fiscale ZVDVFZ62P51D595R, attivita' 3: Accesso negato
Codice Fiscale HVVFPL60A25L248H, attivita' 2: Accesso negato
Codice Fiscale ZVDVFZ62P51D595R, attivita' 3: Accesso negato
Codice Fiscale HVVFPL60A25L248H, attivita' 3: Accesso negato
Codice Fiscale HVVFPL60A25L248H, attivita' 2: Accesso negato
Codice Fiscale ZVDVFZ62P51D595R, attivita' 1: Accesso negato
ZVDVFZ62P51D595R: 0 3 0
LPNVRR74D54A850W: 0 1 0
CNZVVF49M48E182X: 0 1 1
HVVFPL60A25L248H: 3 0 0
```

Ulteriori specifiche

- La lista collegata deve essere definita come tipo di dato astratto, cioè separando interfaccia e implementazione.

- Verificare la correttezza della linea di comando e la corretta apertura dei file; in caso di errore, stampare un messaggio e terminare l'esecuzione.
- Il programma deve essere costituito dai seguenti file:
  - `main.c` contenente (tra eventuali altre) la funzione `main`;
  - `listaUtenti.c` con la definizione delle funzioni su liste (ed eventuali altre);
  - `listaUtenti.h` con le definizioni dei tipi di dato e le dichiarazioni delle funzioni definite in `listaUtenti.c` e utilizzate in `main.c`;
  - `Makefile` che permetta di costruire l'eseguibile con un singolo comando `make`.

Esercizio - Parte 2 (max 5 punti)

Fare in modo che, dopo tre tentativi di accesso negati consecutivi per un utente, la sua tessera venga disattivata, cioè che i successivi tentativi di accesso, indipendentemente dai contatori degli ingressi, vengano negati.