HE Vinci, Session janvier 2024, 09/01/24 de 13h30-15h30

NOM : Login :

PRENOM : Num machine :

BINV2011 : LANGAGE C

EXAMEN JANVIER 2024

ANTHONY LEGRAND, JEROME PLUMAT

Durée de l'examen : 2 h (pas de sortie durant les 60 premières minutes)

Consignes importantes :

- Bootez votre PC sur Ubuntu.
- Utilisez la feuille de login fournie pour vous connecter.
- Notez sur cette page de garde vos nom et prénom, votre login d'examen ainsi que le numéro de votre machine (si disponible).
- Les fichiers de l'examen se trouvent sur votre drive U, vous avez à votre disposition un fichier source et un fichier de données.
- Compilez votre code avec la commande cc, définie avec l'alias précisé si dessous sur les machines de l'IPL :

```
gcc -std=c11 -pedantic -Werror -Wall -Wvla -Wno-unused-variable
```

- Nous vous demandons de faire particulièrement attention à ce que votre code ne produise pas d'erreur de compilation, au risque d'être pénalisé.
- **Travaillez localement sur votre machine** (et non sur le drive U). **À la fin de l'examen, copiez votre fichier *billetterie.c* dans un répertoire à la racine de votre drive U dont le nom suit la nomenclature suivante : NOM_PRENOM.**

  Par exemple : « LEGRAND_ANTHONY »

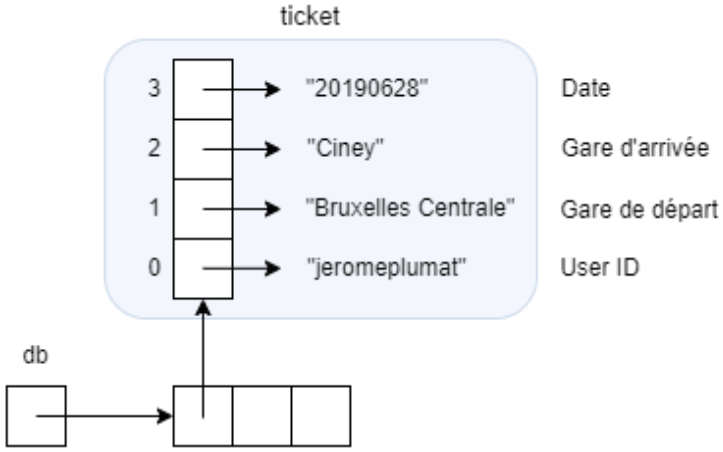
ÉNONCÉ

Il vous est demandé de compléter le fichier source « **billetterie.c** ». Ce dernier doit permettre de gérer une base de données qui contiendra des tickets de train. Le fichier source fourni contient déjà du code et présente une série de fonctions à compléter.

La base de données est construite autour d'une table qui contient au maximum 100 tickets. Cette table a une taille physique fixe durant toute l'exécution.

Pour chaque ticket, on retiendra les informations suivantes dans un tableau (voir exemple ci-contre) :

- L'identifiant de l'utilisateur qui a pris ce ticket (index 0),
- La gare de départ (index 1),
- La gare d'arrivée (index 2),
- La date (index 3).



Toutes les informations sont stockées sous la forme de chaîne de caractères. Les index de chaque information sont donnés en constantes en début de fichier source. Pensez à les utiliser.

Toutes les dates sont stockées sur le même format et leurs chaînes ont toutes exactement la même longueur.

Il vous est demandé d'implémenter les 4 fonctionnalités suivantes (correspondant au menu du programme) :

1. Ajout d'un ticket. Cette opération se fait en trois étapes :
    a) Collecte des informations de l'utilisateur : userID, noms des gares et date (en utilisant la fonction `get_data_from_user()`) ;
    b) Création d'un ticket avec réservation de la mémoire et copies profondes des informations de l'utilisateur (fonction `create_ticket()`) ;
    c) Et enfin ajouter ce nouveau ticket à la db (fonction `add_ticket_to_db()`).
2. Recherche et mise à jour des gares de départ ou de destination d'un ticket. Cette opération se déroule en quatre étapes :
    a) La collecte des informations permettant l'identification du ticket : le userID et la date du voyage (fonction `get_data_from_user()`) ;

   b) La recherche d'un ticket sur base de ces informations ; en cas de doublons, le premier ticket trouvé est renvoyé (fonction `find_ticket()`) ;
   c) Si le ticket est trouvé, il faut demander le/les noms des nouvelles gares ; l'utilisateur peut entrer des chaines de caractères vides, indiquant ainsi que la gare correspondante ne sera pas modifiée (fonction `get_data_from_user()`) ;
   d) La modification des informations du ticket (fonction `update_ticket()`) .

3. Affiche tous les tickets de la db (fonction `print_all_tickets()` qui appelle la fonction fournie `print_ticket()`) .
4. Termine le programme après libération de la mémoire (fonction `free_db()`) .

SCENARIO FOURNI

L'énoncé de l'examen vient avec un fichier `**data**`. Ce fichier contient un scénario vous permettant de rapidement tester l'ensemble des fonctionnalités. Ce fichier effectue les opérations suivantes :

1. Création d'un ticket avec les informations (menu 1) :
   userID : « jplumat »
   gare de départ : « Bruxelles Centrale »
   gare d'arrivée : « Namur »
   date : « 20231128 »
2. Affichage de toute la db (menu 3).
3. Création d'un ticket avec les informations (menu 1) :
   userID : « alegrand »
   gare de départ : « Bruxelles Centrale »
   gare d'arrivée : « Ciney »
   date : « 20231128 »
4. Création d'un ticket avec les informations (menu 1)
   userID : « alegrand »
   gare de départ : « Ciney »
   gare d'arrivée : « Bruxelles Centrale »
   date : « 20231129 »
5. Affichage de toute la db (menu 3).
6. Tentative de modification de ticket avec les informations (menu 2)
   userID : « plumat »
   date : « 20231128 »
   Ticket non trouvé (mauvais userID).
7. Modification de ticket avec les informations (menu 2)
   userID : « jplumat »

date : « 20231128 »

Gare de départ inchangée, nouvelle gare d'arrivée : « Libramont »

8. Affichage de toute la db (menu 3).

   Devrait afficher :

```
1. Add a ticket
2. Modify a ticket
3. Print all tickets
0. Quit
What is your choice ?
>> You wisely choose: 3
Database of tickets:
Ticket:
        ClientID: jplumat
        FROM: Bruxelles Centrale
        TO: Libramont
        Date: 20231128
Ticket:
        ClientID: alegrand
        FROM: Bruxelles Centrale
        TO: Ciney
        Date: 20231128
Ticket:
        ClientID: alegrand
        FROM: Ciney
        TO: Bruxelles Centrale
        Date: 20231129
```

9. Quitter l'application (menu 0).

POINTS D'ATTENTION

Veillez en particulier à :

- Utiliser les fonctions définies et respecter les spécifications lors de leurs implémentations.
- Utiliser la fonction `get_data_from_user()` à chaque fois que vous souhaitez collecter une information de l'utilisateur. On suppose que l'utilisateur ne rentrera jamais plus de `MAX_LENGTH` caractères.
- Dans la base de données de « tickets », toutes les tailles physiques doivent correspondre aux tailles logiques.
- Utiliser le fichier « data » qui vous est fourni avec des informations permettant de tester différents scénarios d'exécution. Vous pouvez utiliser ce fichier avec la commande :

  ```
  ./a.out < data
  ```