ESISAR
CS353 - Algorithmique

TDM 1

1 Objectifs des TDM numéro 1 à 4

Les TDM1 à 4 vont permettre la découverte des arbres binaires et des tables de hachage.
Quatre TDMs sont proposés :
- Le TDM1 pour présenter le contexte et réviser les listes chaînées
- Le TDM2 sur les arbres binaires de recherche standards
- Le TDM3 et 4 sur les tables de hachage

L'évaluation des TDMs se fera sur la base d'un "cahier de laboratoire" par binôme (fichier au format pdf). Dans ce cahier, vous noterez votre avancement, vos analyses et observations, mettrez une copie des codes sources et des captures d'écran des tests exécutés. Un cahier de laboratoire sera rendu après les TDM1 à 4.

Pour les TDM 1,2,3,4 : vous devez utiliser impérativement le langage C et travailler sous Linux (gedit + gcc + makefile).

Pour les TDM 5,6,7,8 : vous devez utiliser impérativement le langage Java et travailler sous Linux

Un travail personnel entre les séances est indispensable pour pouvoir réaliser la totalité des exercices.

2 Le contexte. Implémentation naïve.

2.1 Introduction

Vous êtes le responsable informatique d'une startup de la téléphonie mobile. Votre société a commencé le déploiement de son réseau de téléphonie cellulaire, et vous êtes missionné pour réaliser le système de facturation.

Le problème est le suivant :
- les équipements du réseau alimentent tout au long du mois un fichier de log, qui trace tous les

appels réalisés (une ligne de log contient le numéro de téléphone, la date et le coût de l'appel en centimes d'euros)
•à la fin du mois, le fichier de log est envoyé au service facturation. Ce fichier doit être traité par un programme, qui doit produire un tableau à deux colonnes : la colonne 1 contient les numéros de téléphone, la colonne 2 contient le montant total des communications. Le tableau doit être trié par ordre croissant de numéro de téléphone.

Votre société dispose de 20 000 clients qui réalisent une centaine d'opérations par mois (une opération peut être un appel ou un envoi de SMS). Le fichier de log contient donc environ 2 millions de lignes.

2.2 Questions

Vous allez réaliser une première version naïve du programme, en utilisant une liste simplement chaînée triée.

Question 1

Réalisez une structure `Client` permettant de stocker un numéro de téléphone (sous la forme d'un int), un nombre d'appel, un coût total en centimes d'euros, une référence vers cette structure (structure auto référentielle).

Réalisez une fonction prenant en entrée les 3 premiers paramètres et retournant un pointeur vers une telle structure.

Question 2 – Insertion

Réalisez une fonction qui insère une ligne du fichier de log dans la liste chaînée :

```
struct Client* addLogLine(struct Client* list ,int numero, int prixAppel) ;
```

Le fonctionnement de cette méthode sera le suivant :
•vous naviguez dans la liste pour trouver si le client (identifié par son numéro de téléphone) est présent dans la liste
•si vous trouvez le client : vous incrémentez de 1 le nombre d'appel et vous augmentez le coût total du prix de l'appel
•si vous ne trouvez pas le client, vous l'insérez à la bonne position, et vous le créditez du prix de l'appel et de 1 appel.

La liste chaînée doit être triée en permanence : si vous voulez insérer le client 4, il faut l'insérer entre le client 2 et le client 6.

Question 3 – Test du temps d'éxecution de votre programme

Faire une fonction

```
void dumpList(struct Client* list) ;
```

qui affiche le contenu de toute la liste.

Faites un test du temps d'exécution en utilisant le code suivant :

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

…

// Nombre total de lignes dans le fichier
#define NBLOGLINE    20000000

// Nombre de clients
#define NBCLIENT    20000

int main()
{
    //
    struct Client* list = NULL;
    int i;
    int numeroTel;
    int prixAppel;

    // Aide au calcul du pourcentage d'avancement
    int pas = NBLOGLINE/100;

    printf("****** Facturation appels telephoniques ******\n");


    for(i=0;i<NBLOGLINE;i++)
    {

        // Génération d'un numéro de telephone portable
        numeroTel = 600000000+(rand() % NBCLIENT);

        // Donne un prix d'appel compris entre 0.01 et 4 euros
        prixAppel = (rand() % 400)+1;

        // Ajout de cette ligne de log dans la liste des clients
        list =  addLogLine(list ,numeroTel,prixAppel);

        //
        // printf("numero=%d prix = %d\n",numeroTel,prixAppel);


        // Affichage du pourcentage d'avancement
        if ((i % pas)==0)
        {
             printf("Done  = %d %%...\n",i/pas);
        }
    }

    //
    dumpList(list);


    //
    printf("======= Facturation appels telephoniques ======\n");

    return 0;
}
```