Projet : reconnaissance d'activité humaine via un smartphone.

Il existe beaucoup d'applications permettant à un utilisateur de smartphone de connaître en direct (en temps réel) ses performances sportives (compter son nombre de pas, sa vitesse etc…). Pour cela, l'application a besoin d'identifier le type d'activité de l'utilisateur. L'idée du projet est de créer un programme qui permet de reconnaître l'activité de l'utilisateur à partir de données captées sur son appareil mobile.

1. Idées générales

1.1. Phase 1 : (les fichiers récoltés vous seront fournis)

Sur des cobayes consentants, le système a récupéré des données à partir de capteurs installés sur leur mobile (notamment leur géolocalisation). Vous aurez ainsi beaucoup d'exemples, comme sur le schéma ci-dessous. Pour chacun de ces exemples, vous connaîtrez le type d'activité correspondante puisque c'est la phase « récolte d'informations ».

x	x	x	x	x	…	x	x	x	downstairs
x	x	x	x	x	…	x	x	x	downstairs
…									
x	x	x	x	x	…	x	x	x	upstairs
x	x	x	x	x	…	x	x	x	downstairs
…									
x	x	x	x	x	…	x	x	x	jogging
x	x	x	x	x	…	x	x	x	walking
…									

1.2. Phase 2 : modéliser et évaluer

A partir de toutes ces données, vous établirez un « pattern » pour chaque type d'activité. Pour analyser la performance de votre programme, vous aurez besoin de réaliser quelques statistiques. C'est pourquoi, vous allez commencer par créer une librairie « classificationStatistics » (cfr instructions ci-dessous). Lorsque votre programme sera prêt, vous pourrez passer en phase 3.

1.3. Phase 3 : utiliser

Ainsi, lorsqu'un utilisateur se servira de l'application sur son mobile, votre programme sera capable de déterminer l'activité de celui-ci.

2. Bibliothèque « classificationStatistics » :

Cette bibliothèque se veut généraliste pour des statistiques concernant des classifications. L'utilisateur de votre librairie n'aura accès qu'aux fonctions d'affichage précisées en 2.1 / 2.2 et 2.3. Ainsi, ce sont les trois seules fonctions publiques de votre librairie.

Voici le type de statistiques nécessaires dans ce cas. A la sortie de votre programme, vous aurez :

Classes réelles	Classes estimées	Résultat
downstairs	downstairs	ok
downstairs	upstairs	Ko
…		
jogging	jogging	Ok
walking	jogging	Ko

Parfois, une classe correspond à une valeur entière. Partons de cette idée.

(**) Voyez s'il est possible (et comment) que chaque fonction puisse prendre, chaque fois que nécessaire, les arguments dans un format chaines de caractères ou entiers en fonction de ce que renvoie le programme. Pensez aux pointeurs génériques.

Vous testerez vos différentes fonctions en mettant dans le programme principal :

```
int realClasses[8] = { 5, 2, 5, 3, 5, 3, 2, 4 };
int estimateClasses[8] = { 5, 5, 1, 2, 1, 3, 2, 4 };
int nbTests ; // nbre de valeurs dans les tableaux ci-dessus.
```

par exemple.
NB : les noms de variables ou de fonctions en italique souligné sont à reprendre tels quels.

2.1. Affichage des résultats par classes

Créez une fonction *displayResultsForEachClass(realClasses, estimateClasses, nbTests)* permettant d'obtenir l'affichage suivant :

```
classe      |       bien classes   |       total   |      Pourcentage
  5         |              1       |         3     |           33.33%
  2         |              1       |         2     |           50.00%
  3         |              1       |         2     |           50.00%
  4         |              1       |         1     |          100.00%
```

NB :

- C'est le hasard si les pourcentages sont affichés par ordre croissant.
- N'oubliez pas de respecter les règles de clean code notamment votre découpe doit scinder l'affichage et le calcul (évitez les effets de bord).

2.2. Affichage du pourcentage global de bon classement

Créez une fonction permettant d'afficher « L'accuracy est de 78% » par exemple. La fonction sera nommée *displayAccuracy(realClasses, estimateClasses, nbTests).*

2.3. Affichage par classe

Créez une fonction *displayClass(realClasses, estimateClasses, nbTests)* qui affiche, par classe existante, un titre reprenant le numéro et le libellé de la classe, l'affichage des classements correspondants. Ainsi, sur l'exemple ci-dessous, le programme aurait dû classer Jogging 10 et, sur ces 10 fois, le programme a classé Jogging 5 fois, Marche 2 fois etc…

A classer		1	2	3	….
libelle	nombre				
Jogging	10	5	2	1	…
Marche	5				
…					

Attention, une analyse préalable est nécessaire pour factoriser votre code au mieux.

2.4. A rendre

Remettre sur Moodle un fichier pdf dans lequel vous aurez répondu aux questions suivantes :

1. Créez un schéma des fonctions nécessaires grâce auquel on peut voir que telle fonction appelle telle fonction ;
2. Présentez les structures de données que vous utiliserez ;
3. Pour chacune des fonctions du travail, indiquez les spécifications à savoir
    a. Indiquez son nom ;
    b. Indiquez les paramètres d'entrée – avec leur type et un éventuel commentaire s'il est nécessaire à la compréhension ;
    c. Idem paramètre de sortie ;
    d. En une phrase, précisez ce que fait la fonction.
4. Organisez-vous pour vous partager le travail. Prévoyez que le codeur d'une fonction n'en soit pas le testeur. NB : si votre analyse est suffisamment approfondie, cela ne posera aucun problème.

Quelques préconditions :

- vous pouvez considérer que vous aurez maximum 10000 tests et 20 classes

NB :

- toutes les classes sont représentées dans realClasses
- utilisation de pointeurs non nécessaire (pour la proposition de base)
- Soyez attentifs à la complexité (les tableaux realClasses et estimateClasses peuvent contenir 10000 cellules).
- Liste chainée ou tableau ?