TP 1
LISTE D'ATTENTE

Introduction

Pour le premier tp, vous devez construire un logiciel C++ qui classera les patients d'une salle d'urgence. Ce logiciel va prendre en entrées une liste de patients avec la sévérité de leur problème et le nombre de minutes écoulé depuis leurs arrivées. En sortie, le logiciel va produire une liste classée des patients afin de maximiser le nombre de patients qui seront vus en temps règlementaire par le personnel médical.

Description des entrées

Le programme C++ que vous allez construire va être lancé en ligne de commande sur les serveurs malt de labunix. Le logiciel va recevoir un argument obligatoire qui sera possiblement suivi d'un argument optionnel. L'argument obligatoire est le nom complet d'un fichier, ce fichier contiendra la liste des patients. L'argument optionnel est une valeur numérique qui indiquera le temps moyen d'une consultation (tc) avec le personnel médical. Cette valeur sera égale à 5 (minutes) par défaut (si aucune valeur n'est donnée). Votre logiciel doit vérifier si le fichier de patient existe. Aussi, si l'argument optionnel est donné, il doit être une valeur entière de 1 ou plus. S'il y a une erreur, votre logiciel doit terminer avec un message d'erreur sur le canal `err` et doit retourner le code -1.

Le fichier contenant la liste des patients sera un fichier texte simple. Il contiendra la description d'un patient par ligne. Un patient (i) est décrit par trois valeurs numériques entières : un numéro de patient ($1 \leq n_i$), le temps ($1 \leq t_i$) écoulé depuis son arrivée (en minute), et sa priorité ($2 \leq p_i \leq 5$).

Exemple du contenue d'un fichier :

```
43525 5 2
25545 7 5
7455 3 4
```

Il y a trois patients, le premier ($n_0 = 43525$) attend depuis $t_0 = 5$ minutes et à une priorité de $p_0 = 2$. Le second ($n_1 = 25545$) attend depuis $t_1 = 7$ minutes et à une priorité de $p_1 = 5$ et finalement, le troisième ($n_2 = 7455$) attend depuis $t_2 = 3$ minutes et a une priorité de $p_2 = 4$.

Classement

Votre logiciel doit classer les patients dans l'ordre où ils vont consulter le personnel. Cet ordre est déterminé par deux facteurs. Le premier facteur est leurs temps d'arrivée et le deuxième est la maximisation du nombre de patients vus dans les temps règlementaires.

Temps d'arriver

Le premier facteur est important. Il suit une règle simple, si deux patients (i et j) ont la même priorité $p_i = p_j$, alors le patient qui est arrivé en premier, c'est-à-dire celui qui a le temps écoulé le plus grand, doit passer en premier. Donc, si $p_i = p_j$ et $t_i > t_j$ alors, i passe en premier.

Nombre de patients vu en temps règlementaire

Le deuxième facteur est une métrique que vous allez calculer sur les résultats. Vous devez trouver un algorithme qui va donner une bonne valeur (espérons qu'elle soit optimale) pour cette métrique.

La table suivante présente le temps conseillé par le gouvernement pour qu'un patient soit vu selon la sévérité de sa condition.

Priorité (sévérité)	Temps d'attente maximum (en minute)
1	0
2	15
3	30
4	60
5	120

Il est à remarquer que la priorité de 1 n'est donnée qu'à titre indicatif, elle n'apparaitra pas dans le fichier d'entrées. Les patients ne devraient pas avoir de temps d'attente supérieur au temps indiqué dans cette table, bien que préférables, cela n'arrive pas toujours. Votre logiciel doit classer les patients afin de limiter le nombre de patients vu en retard. Par exemple, un patient dont le t_i est de 10 minutes (cela fait 10 minutes qu'il attend) et qui a une priorité de $p_i = 3$ devrait passer dans les 20 prochaines minutes (30 - 10 = 20). Si le temps de consultation est $tc = 5$, alors nous avons 20 minutes, divisé par 5 minutes par patient, cela nous donne 4 patients maximum devraient être placé avant lui, sans pour autant contredire la première règle. Votre algorithme doit toujours respecter la première règle, et doit maximiser le nombre de patients respectant la seconde. Pour mesurer le nombre de patients respectant la seconde règle, nous allons utiliser le *fractile*.

Fractile

Votre logiciel va devoir évaluer une métrique pour mesurer ces performances. Disons que nous avons placé les patients dans l'ordre voulu, nous pouvons construire la table suivant :

Priorité (p)	Nombre de patients	À temps	En retard	Fractile
2	10	2	8	0.2
3	20	8	12	0.4
4	25	16	9	0.64
5	25	22	3	0.88

Le fractile est calculé en divisant le nombre de patients à temps dans une catégorie (priorité) par le nombre de patients de cette catégorie. Finalement, pour avoir un pointage unique s, nous prenons la moyenne géométrique des fractiles. Pour calculer une moyenne géométrique, nous commençons par multiplier les valeurs ensemble. Dans notre exemple $m = 0.2 * 0.4 * 0.64 * 0.88 = 0.045056$. Ensuite nous prenons la k ième racine du résultat. Pour notre projet, $k = 4$ en tout temps. Donc, le pointage unique de notre exemple $s = \sqrt[4]{0.045056}$ = 0.46 (remarquez que $\sqrt[4]{x} = x^{0.25}$).

Production des sorties

Finalement, votre logiciel doit afficher les résultats : les patients dans l'ordre assigné par le logiciel et une table de fractile. La liste des patients est affichée à raison de 1 patient par ligne. Pour chaque patient, vous affichez sa position dans la liste (le premier patient de la liste est à la position 1), 1 espace, son numéro de patient n_i, 1 espace et sa priorité p_i.

Lorsque tous les patients ont été affichés, vous afficher une ligne contenant 8 tirets : --------. Sur les lignes suivantes, vous affichez la priorité et le fractile pour chaque catégorie, séparé de 1 espace. Finalement, sur la dernière ligne, vous affichez la moyenne géométrique.

Directive

1. Le tp est à faire seul ou en équipe de deux (maximum).
2. Commentaire :
    a. Commentez l'entête de chaque fonction. Ces commentaires doivent contenir la description de la fonction et le rôle de ces paramètres.
    b. Une ligne contient soit un commentaire, soit du code, pas les deux.
    c. Utilisez des noms d'identificateur significatif.
    d. Utilisez le français.

3. Code :
    a. Pas de `goto`, `continue`.
    b. Les `break` ne peuvent apparaître que dans les `switch`.
    c. Un seul `return` par méthode.
4. Indentez votre code. Assurez-vous que l'indentation est faite avec des espaces.

Remise

Remettre le tp par l'entremise de Moodle. Remettez votre fichier `'*.cpp'`, nous utiliserons un seul fichier de code pour le premier tp. Le tp est à remettre avant le 22 février 23:59.

Évaluation+

- Fonctionnalité (8 pts) : Votre tp doit compiler sans erreur (il peut y avoir des warnings). J'utilise la ligne de compilation suivante : `g++ nomTp.cpp -lm -std=c++11`
- Structure (2 pt) : Il faut avoir **plusieurs** fonctions. Construisez un code bien structuré.
- Lisibilité (4 pts) : commentaire, indentation et noms d'identificateur significatif.
- Compétition (1 pt) : Les tps vont être classées en ordre de performance sur la moyenne géométrique. L'équipe avec le meilleur pointage aura 1 pt, les autres auront un ratio proportionnel à leur position dans la liste de pointage.