Tâche 1.2 – Des moutons à foison (sheep) (100 pts)

Azamat le fermier participe à un jeu télévisé, et à côté des divertissements habituels avec chanteurs en playback, il peut aussi gagner de nouveaux moutons pour sa ferme.

Le jeu se déroule de la manière suivante. Les moutons qu'Azamat peut gagner sont introduits un à un dans le studio. Si Azamat peut alors dire combien de paires de moutons sont dans le mauvais ordre parmi tous les moutons arrivés jusque là dans le studio (c'est-à-dire le nombre de paires où le mouton arrivé le plus tôt est plus grand que le mouton arrivé plus tard), il gagne aussi ce nouveau mouton. S'il ne donne pas de réponse ou ne répond pas dans les temps, le jeu s'arrête.

Azamat ne sait pas très bien compter, mais grâce à sa considérable expertise en moutons, il est capable de déterminer en un clin d'œil la taille exacte d'un mouton. Il a conclu un pacte avec vous : il vous informe de la taille de chaque nouveau mouton, et vous lui donnez la réponse correcte au jeu. Si Azamat gagne tous les moutons, vous recevrez pour les 3 années à venir un quart des revenus issus de leur laine !

Tâche

Aidez Azamat à gagner le jeu télévisé.

Limites et contraintes

- $1 \leq N \leq N_{MAX}$, le nombre total de moutons ;
- $0 \leq M \leq M_{MAX}$, la taille maximale d'un mouton, dans une unité de longueur quelconque ;
- Tous les moutons ont une taille différente.

	N_{MAX}	M_{MAX}
Sous-tâche A (12 pts)	200	1 000
Sous-tâche B (18 pts)	3 000	100 000
Sous-tâche C (50 pts)	50 000	1 000 000
Sous-tâche D (20 pts)	50 000	10^{12}

Durée maximale d'exécution : **3 secondes**. Limite mémoire : **512 MB**.

Fonction à implémenter

C++ : `long long newSheep(long long N, long long M, long long i, long long size)`
Java : `long newSheep(long N, long M, long i, long size)`
Cette fonction prend comme arguments :

N Le nombre total de moutons

M La taille maximale d'un mouton

i Le numéro du mouton, donc dans des appels successifs 1, 2, 3, …

size La taille du nouveau mouton

Cette fonction doit donner comme résultat un entier : la réponse la réponse à la question du jeu après l'arrivée du mouton.

Exemple

Voici un scénario possible :

N	M	i	$size$	Moutons déjà présents	Résultat	Paires dans le *mauvais ordre*
4	10	1	8		0	
4	10	2	5	8	1	$(8, 5)$
4	10	3	7	8, 5	2	$(8, 7)$, $(8, 5)$
4	10	4	2	8, 5, 7	5	$(8, 7)$, $(8, 2)$, $(5, 2)$, $(8, 5)$, $(7, 2)$

Remarques pratiques

Cette tâche est interactive. Vous ne pourrez donc pas utiliser la compilation et le test automatique de gedit pour cette tâche. Suivez les étapes ci-dessous pour compiler et tester votre programme manuellement. Demandez de l'aide à un surveillant si nécessaire.

- Ouvrez un terminal (l'application LXTerminal)
- Pour obtenir la liste des fichiers dans le dossier courant, utilisez la commande `ls`
- Pour changer de dossier et aller vers le squelette de cette tâche, utilisez `cd nom_du_dossier`

Pour ceux qui utilisent C++, les commandes sont les suivantes :

- Compilation : `g++ -std=c++11 -Wall -Wextra -Wshadow grader.cpp sheep.cpp`
- Exécution : `./a.out < input.txt`

Pour ceux qui utilisent Java, les commandes sont les suivantes :

- Compilation : `javac grader.java`
- Exécution : `java grader < input.txt`

Les résultats des appels successifs à votre fonction seront affichés dans la console sur des lignes séparées.

Remarques

— Vous devez uniquement soumettre le fichier `sheep.cpp` ou `sheep.java`. Vous devez implémenter la fonction `newSheep`, comme décrite ci-dessus.
— N'imprimez rien à la sortie standard dans le programme que vous soumettez.