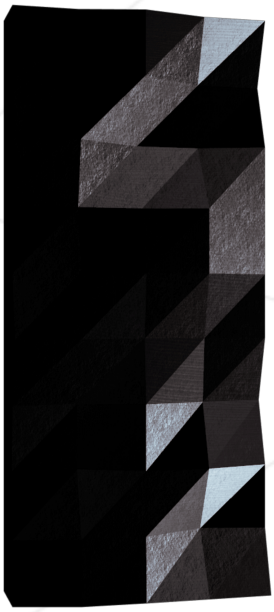

Rush d'Algo

FactRace

42 staff staff@42.fr

Résumé: Ce rush est un concours de rapidité pour vous faire réfléchir sur des algorithmes et techniques de programmation efficaces.

Table des matières

Chapitre I

Préambule

En travers de l'histoire qui se répète, l'être humain s'est peu à peu affranchi des servitudes naturelles qui entravaient le plein épanouissement de son caractère fondamental, et le développement de son identité culturelle terrienne fondamentale. Ainsi, en l'an 2015 de notre ère, fut inventé une armure de combat robuste et courageuse, elle-même abritant un combattant robuste et courageux, répondant à ces mots : *"Ne tuez pas l'alien, Emily Blunt s'en chargera."*

FIGURE I.1 – Le combattant robuste et courageux

Chapitre II

Sujet

II.1 Ce-quoi-donc-qui-faut-il-faire

Finalement, c'est pas très compliqué de faire un algorithme pour casser les clefs RSA. Il suffit d'avoir un gros chiffre, puis de simplement chercher ses facteurs premiers. Non, la difficulté, c'est de faire cela rapidement. Et alors là, il y a une lutte acharnée entre les *NSA*, *CIA*, *Apple*, et tout un tas d'autres qui existaient déjà quand vous êtes nés. C'est une course où tout le monde veut griller les autres : la **FactRace**.

Participez-donc à la FactRace. Pour cela, il vous suffit de réaliser un programme qui va faire deux choses en même temps : d'un coté emmagasiner des informations de type "gros_nombre_non_premier", beaucoup d'information et de difficulté croissante (plus ou moins), et de l'autre afficher dans un temps donné la décomposition "gros_nombre_non_premier=facteur_1*facteur_2" situé le plus loin possible dans la liste reçue.

- Votre programme s'appelle **factrace**.
- Votre programme prend un parametre : une durée en millisecondes.
- Votre programme lit sur l'entrée standard les informations.
- Votre programme affiche le gros nombre suivi de sa décomposition selon l'affichage vu ci-dessus, terminé par un "\n".

Le format des infos sur l'entrée standard est d'un nombre en base 10 par ligne :

```
4
12
6126755
229039
3037051
11915666136247
5064029346065
691766920870797
1148183999166558
[...]
```

Votre affichage se fait sur la sortie standard.
Vous pouvez afficher autant de réponses que vous le souhaitez, seule la dernière sera prise en compte pour déterminer jusqu'où vous êtes allé dans la liste.

La décomposition doit être exactement en 2 facteurs (pas davantage), et aucun des facteurs ne peut être égal à 1.
Losrque le temps est écoulé, votre programme est killé. Prévoyez une certaine marge et assurez-vous d'avoir au moins affiché une réponse. Il est conseillé de faire des *write* d'un seul bloc.

Par exemple, en réponse aux nombres précédents, votre programme peut afficher :

```
4=2*2
12=3*4
691766920870797=3*230588973623599

```

ou bien directement :

```
1148183999166558=2*574091999583279

```

Parmi les programmes qui fonctionnent correctement, celui qui affiche la décomposition le plus loin dans la liste a 125, le suivant 120, ainsi de suite. Le seuil de validation est à 1. Votre performance détermine l'XP reçue. La vitesse et l'optimisation de votre programme sont donc cruciaux. L'écart ici présenté de 5 points entre chaque rang est suceptible de varier selon le nombre de groupes engagés, afin de répartir les notes des participants de façon homogène sur une large plage pour assurer un véritable enjeux durant la compétition.

Ce rush se fait à 2.

P.S. : si vous pensez que c'est compliqué, attendez d'avoir à vraiment cracker des clefs RSA.

II.2 Rendu

- Vous devez rendre, à la racine de votre dépôt de rendu, un fichier `auteur` contenant vos 2 login chacun sur une ligne et suivi d'un '\n' :

```
$>cat -e auteur
xlogin$
ylogin$
$>
```

- Vous devez avoir un `Makefile` avec les règles usuelles. Dans le doute, mettez toutes les règles connues.
- Seul le contenu présent sur votre dépot sera évalué.

II.3 Fonctions autorisées

- read
- write
- malloc
- free
- strerror
- la directive de compilation `__asm__`
- une librairie de gestion des grands nombres

Chapitre III

Consignes

- Une moulinette est à disposition. Utilisez-la. Elle vous permettra de vous comparer entre vous.
- La note sera obtenue par la moulinette de rapidité, si les résultats sont corrects.
- Votre projet doit être en C à la Norme. La Norminette ne sera pas utilisée pour vérifier la Norme qui s'applique donc dans son ensemble.
- Vous devez gérer les erreurs de façon sensible. En aucun cas votre programme ne doit quitter de façon inattendue (Segmentation fault, bus error, double free, etc).
- Toute mémoire allouée sur le tas doit être libérée proprement.
- Vous ne pouvez pas utiliser votre bibliothèque `libft`.

Bon courage à tous pour ce rush !