420KBG – Activité 02

Il est temps de vous donner un peu plus de latitude pour étendre vos ailes et réfléchir à un design qui vous ressemblera. Pour cette raison, nous nous engageons dans une activité par étapes, mais où la progression d'étape en étape ne vous sera révélée qu'en temps et lieu.

Il est donc impératif de réfléchir avant de mettre les mains sur le clavier, et de produire du code propre, modulaire et modifiable… car qui sait ce qui nous attend?

C'est un peu comme la « vraie vie », quoi ☺

Idée générale

Votre mission est d'offrir une `API` exposant un ensemble de services de manipulation de chaînes de caractères qui servira à des multitudes (!) de programmeuses et de programmeurs dans le monde. Notez que les chaînes de caractères ne se limiteront pas à du texte « lisible par des humains ».

Quelques définitions

Pour les fonctionnalités détaillées ici :

- Une **séquence à demi-ouverte** sera telle que l'indice de début sera inclus et l'indice de fin sera exclu, donc une paire $[début, fin)$
- Les **indices débuteront à zéro**
- Un **blanc** ne se limitera pas au seul symbole `' '` et pourra comprendre sauts de ligne, tabulations, etc. Il existe sûrement des services standardisés pour vérifier si un caractère est un blanc ou non
- Un **mot** sera une séquence de non-blancs délimitée par des blancs, ou se trouvant aux extrémités d'une chaîne de caractères

Services à offrir

Cette gamme de services comprendra les fonctionnalités suivantes.

Un service acceptant une chaîne de caractères s et un entier n, et qui retournera l'indice de la première séquence de n éléments consécutifs de même valeur dans s, ou -1 si aucune telle séquence n'existe, ou encore si $n \leq 0$

Un service acceptant une chaîne de caractères s, et qui retournera une paire dont le premier élément sera l'indice du début de la plus longue séquence de caractères identiques consécutifs trouvés dans s, et le second élément sera l'indice de la fin de cette « sous-séquence ». Veillez à ce que la paire d'indices retournée forme une séquence à demi-ouverte. Si s est vide, alors retournez une paire $[fin, fin)$. Si plusieurs séquences distinctes ont la même (plus longue) longueur, alors retournez une paire $[début, fin)$ sur la première d'entre elles.

Un service acceptant une chaîne de caractères s, et qui retournera une chaîne de la même taille que s, mais dont le contenu est l'inverse de l'ordre des mots dans la chaîne s sans toutefois modifier l'ordre et la taille des séquences de blancs dans s. Par exemple :

- si s est `" j'aime mon prof "`, alors le service retournera `" prof mon j'aime "`

- si *s* est `"yo"`, alors la fonction retournera `"yo"`
- si *s* est `"  yo man"`, alors la fonction retournera `"  man yo"`
- si *s* est `"yo  man"`, alors la fonction retournera `"man  yo"`

Un service acceptant une chaîne de caractères *s*, et qui retournera une chaîne de la même taille que *s*, mais dont le contenu est tel que seul l'ordre des symboles dans chacun des mots est inversé. Par exemple :

- si *s* est `" j'aime  mon prof "`, alors la fonction retournera `" emia'j  nom forp "`
- si *s* est `"yo"`, alors la fonction retournera `"oy"`
- si *s* est `"  yo man"`, alors la fonction retournera `"  oy nam"`
- si *s* est `"yo  man"`, alors la fonction retournera `"oy  nam"`

Un service acceptant une chaîne de caractères *s* et un caractère *c*, et qui retournera une chaîne de caractères équivalente à *s*, mais dont toutes les occurrences de *c* au début auront été supprimées.

Un service acceptant une chaîne de caractères *s* et un caractère *c*, et qui retournera une chaîne de caractères équivalente à *s*, mais dont toutes les occurrences de *c* à la fin auront été supprimées.

Un service acceptant une chaîne de caractères *s* et un caractère *c*, et qui retournera une chaîne de caractères équivalente à *s*, mais dont toutes les occurrences de *c* au début et à la fin auront été supprimées.

Un service acceptant une chaîne de caractères *s*, et qui retournera une chaîne de caractères équivalente à *s*, mais dont toutes les séquences de blancs seront remplacées par un seul caractère d'espacement.

Un service acceptant une chaîne de caractères *s*, et qui retournera une chaîne de caractères équivalente à *s*, mais dont les « mauvais mots » auront été censurés. Le terme « censuré » signifie ici « remplacé par quelque chose de présumé non-offensant ».

Pour les services de censure, démarrez l'exécution avec une courte liste de mots que vous estimez inappropriés

Un service acceptant au moins une chaîne de caractères sans blancs, et qui ajoutera chacune de ces chaînes de caractères la liste des « mauvais mots »; ajouter un mauvais mot existant à la liste des mauvais mots ne doit pas être une erreur.

Un service acceptant une chaîne de caractères *s* sans blancs, et qui supprimera *s* de la liste des « mauvais mots » si elle s'y trouvait au préalable.

Un service permettant d'obtenir la liste des « mauvais mots » connus du serveur. De manière optionnelle, ces mots peuvent être triés en ordre lexicographique croissant, en ordre lexicographique décroissant, en ordre croissant de longueur ou en ordre décroissant de longueur.

Première étape

Note : pour ce qui suit, nous présumerons un service mathématique comme celui de l'activité `00` et dont la version sera v. `1`. Le numéro de version fera partie de l'`URI` du service.

Pour cette première étape, votre travail est de réfléchir à votre `API`. Pour ce faire, envisagez que vous offrirez un service d'aide (p. ex. : `/api/maths/1?`), et que cette première étape vous demande de remettre ce que retournerait un appel à ce service.

Le service d'aide devra retourner un objet `JSON` contenant un tableau de services, donc chaque élément sera un objet `JSON` décrivant le service de manière découvrable par programmation. Les champs clés attendus dans ces objets sont :

- Le champ `authors`, qui permettra d'identifier les auteurs du service
- Le champ `version`, indiquant la version du service. Notez que la valeur de ce champ devrait correspondre à ce qui apparaît dans les `URI` de votre `API`
- La liste des services offerts par votre `API`. Pour chaque service, je m'attends à voir :
- Un champ `method` indiquant le verbe `http` à utiliser
  - Un champ `name` indiquant un nom lisible par une humaine ou par un humain pour le service
  - Un champ `uri` indiquant l'`URI` du service
  - Un champ `role` indiquant le rôle du service, dans une chaîne de caractères lisible par une humaine ou par un humain
  - Un champ `examples` contenant une liste d'appels correct ou incorrects, et de résultats (dans le cas de succès) ou d'erreurs (dans le cas d'échecs) correspondant à ces appels. Notez que l'intention ici est de faire en sorte que, si on sollicite le service par l'`URI` indiquée, on obtiendra le résultat ou l'erreur qui lui correspondra

Exemple concret

Par exemple, prenant les services d'addition et de multiplication de l'activité `00`, qui sont des services `GET`, je m'attendrais à quelque chose comme ceci (le champ `example` de chaque service pourrait être plus étoffé) :

```
{
   "authors" : [
      { "name" : "auteur0" },
      { "name" : "auteur1" }
   ],
   "version" : 1.0,
   "services" : [
      {
         "method" : "get",
         "name" : "help",
         "uri" : "/api/maths/1",
         "role" : "get this help message",
         "examples" : [
            {
```

```
            "uri" : "/api/maths/1?"
         }
      ]
   },
   {
      "method" : "get",
      "name" : "addition",
      "uri" : "/api/maths/1",
      "role" : "compute and return the sum of two numbers",
      "args" : [
         { "name" : op, "value" : '+' },
         { "name" : x, "type" : double },
         { "name" : y, "type" : double },
      ],
      "examples" : [
         {
            "uri" : "/api/maths/1?op='+'&x=2&y=3",
            "result" : 5
         },
         {
            "uri" : "/api/maths/1?op='+'&x=2.5&y=3",
            "result" : 5.5
         },
         {
            "uri" : "/api/maths/1?op='+'&x=&y=3",
            "error" : 400
         }
      ]
   },
   {
      "method" : "get",
      "name" : "multiplication",
      "uri" : "/api/maths/1",
      "role" : "compute and return the product of two numbers",
      "args" : [
         { "name" : op, "value" : '*' },
         { "name" : x, "type" : double },
         { "name" : y, "type" : double },
      ],
      "examples" : [
         {
            "uri" : "/api/maths/1?op='*'&x=2&y=3",
            "result" : 6
         },
         {
            "uri" : "/api/maths/1?op='*'&x=2.5&y=3",
            "result" : 7.5
         },
```

```
            {
               "uri" : "/api/maths/1?op='*'&x=&y=3",
               "error" : 400
            }
         ]
      },
      {
         "method" : "get",
         "name" : "est premier",
         "uri" : "/api/maths/1",
         "role" : "computes and returns whether an integral is a prime",
         "args" : [
            { "name" : op, "value" : 'p' },
            { "name" : n, "type" : integer },
         ],
         "examples" : [
            {
               "uri" : "/api/maths/1?op='p'&n=5",
               "result" : true
            },
            {
               "uri" : "/api/maths/1?op='p'&n=6",
               "result" : false
            },
            {
               "uri" : "/api/maths/1?op='p'&n=1.5",
               "error" : 400
            }
         ]
      }
   ]
}
```

Notez que pour `type` dans les paramètres, les seules options seront `bool`, `integer`, `string` et `double`, le tout dans le but de préserver la simplicité de l'`API`. Il est raisonnable de s'attendre à ce que le serveur valider les « types » annoncés : si un service comme celui nommé `"est premier"` demande que n soit entier, alors il devrait signaler une erreur si n ne respecte pas cette contrainte.

Pour certains des services demandés, le verbe `GET` n'est pas adéquat, et vous voudrez adapter le format ci-dessus (p. ex. : utiliser `"post"` ou `"delete"` à titre de verbe, enlever la section `"args"`, étoffer la section `"examples"`, etc.).

Modalités de remise

Organisation :	Travail individuel ou par équipe de deux
Date de remise :	**Vendredi le 23 octobre 2020 à 23 h 59**
Remise par Colnet?	Oui, dans une archive zip nommée comme suit : `Nom-Prénom.zip` (p. ex. : `Tromblon-Gaetan.zip`) …ou encore, dans le cas d'une équipe de deux `Nom0-Prénom0--Nom1-Prénom1.zip` Cette archive doit contenir : • Un document (.pdf, .docx ou simplement un document texte) dans lequel se trouvera la version « stringifiée » de l'objet `JSON` attendu lors d'un appel à la fonction « help » de votre API • **Attention : ceci est un contrat auquel vous serez lié(e)s par la suite, alors prenez ce travail *très* au sérieux!**

Notez que bien que la remise se limite à une documentation `JSON` de l'`API` demandée, vous pouvez commencer à implémenter tout de suite cette `API` car son implémentation vous sera demandée ultérieurement.

Amusez-vous bien!