Jusqu'à présent, nous nous sommes interessés à des protocoles à caractère textuel. On peut aussi concevoir des protocoles où les requètes et les réponses ont un format non pas textuel mais binaire.

Exercice 1. Requêtes/réponses sous forme de n-uplets.

Dans un premier temps, reprenons l'idée du protocole clé/valeur, mais dans le cas particulier où les clés et les valeurs sont des entiers. Autrement dit, le serveur contient un tableau pour représenter sa base de données. La clé est l'indexe dans le tableau. L'élément à cet indexe du tableau est la valeur. On veut permettre les opérations suivantes :

`PUT(i,k)` mettre la valeur k en position i du tableau
`GET(i)` obtenir la valeur en position i du tableau
`PUTN(i1,k1,...,in,kn)` affecter simultanément n positions du tableau (n est variable)
`GETN(i1,...,in)` obtenir simultanément les valeurs de n positions du tableau (n est variable)

Dans un protocole binaire, les messages (requêtes et réponses), consistent en un certain nombre de champs et chaque champ a un format binaire bien déterminé (un certain nombre d'octets). Pour la première étape de la conception de notre protocole binaire, nous allons devoir décider des champs nécessaires pour chaque requête et chaque réponse possible. Nous représenterons un message par un n-uplet de champs :

```
(champ1,champ2,...,champN)
```

Autrement dit, le n-uplet ci-dessus représente une sequence de n champs. Proposez une représentation sous forme de n-uplet pour chaque requête et chaque réponse possible. Attention : chaque message doit contenir suffisamment d'information pour permettre à son destinataire de déterminer combine de champs il contient.

Exercice 2. Automate du serveur.

Nous allons maintenant concevoir un automate pour le serveur. Comme auparavant, les nœuds de l'automate représentent les différents états du serveur, et les arcs représentent les transitions possibles entre ces états. Ces transitions seront annotées par des conditions, des messages lus ou envoyés (sous la forme des n-uplets de l'exercice précédents), et des actions à effectuer. On préfixera les envois du client par `C:` et ceux du serveur par `S:` pour faciliter la compréhension de l'automate. Si le serveur lit 2 champs d'un message envoyé par le client, on écrira sur la transition correspondante :

```
C:(champ1,champ2)
```

Notez que si ça n'est pas un message complet, alors il faudra ensuite d'autres transitions pour lire les champs restants du message.

Exercice 3. Format binaire des requêtes/réponses.

Dans cet exercice, nous allons prendre les n-uplets conçus dans l'exercice 1 et leur choisir une représentation binaire. Nous utiliserons pour cela les schémas sous forme de "boites" que nous avons vu en cours pour décrire les formats des segments UDP et TCP.

Exercice 4. Valeurs de taille variable.

Supposons à présent qu'on veuille développer un serveur où le tableau ne contient plus des entiers mais plutôt des enregistrements correspondants à la définition de type suivante :

```
struct personne
{
   char* nom;
   char* prenom;
   int datenais;
}
```

On veut permettre les même opérations que précedemment. Concevez un format binaire pour les requêtes/-réponses de ce protocole :

- donnez d'abord les n-uplets
- proposez ensuite pour ceux-ci un format binaire