Niveau 2 en informatique
INF 2054
Pratique /30

Dr. Azanzi Jiomekong

La base de données alimentaire est une plateforme utilisée pour collecter des données sur les aliments et leurs valeurs nutritionnelles (composition en micronutriments et macronutriments) (Wikipedia). De plus, il peut contenir des effets positifs et négatifs sur la santé associés aux constituants alimentaires. En Afrique en général, et au Cameroun en particulier, ces plateformes manquent. Ainsi, le but de cet exercice est de construire une telle plateforme. À cette fin, des données devraient être recueillies par les élèves sur ce qu'ils mangent au quotidien, structurées et utilisées pour construire une base de données sur la composition des aliments. La plate-forme globale devrait détailler les informations sur les composants alimentaires importants sur le plan nutritionnel et fournir des valeurs pour l'énergie et les nutriments, y compris les protéines, les glucides, les lipides, les vitamines et les minéraux, ainsi que pour d'autres composants alimentaires importants tels que les fibres. Un exemple : https://fdc.nal.usda.gov/. Etant donné que la connaissance de la composition chimique des aliments est le premier élément essentiel dans le traitement diététique des maladies, cette plateforme devrait permettre aux utilisateurs de choisir ce qu'ils veulent manger au quotidien.

D'autre part, le transport urbain dans de nombreux pays africains en général et à Yaoundé en particulier est maintenant bien connu. En effet, l'expansion des villes comme Yaoundé s'accompagne d'une croissance territoriale puisque la population occupe plus d'espaces. Ainsi, les distances ne cessent d'augmenter entre les habitants et leur lieu de travail. Par conséquent, il y a une augmentation de l'utilisation des installations existantes et un plus grand nombre de déplacements. Le développement des transports urbains et l'amélioration des services publics, notamment le transport de voyageurs, influencent le temps et le budget de la population. La complexité des différents modes de transport de personnes à Yaoundé et les difficultés à identifier et bien décrire l'endroit où aller aggravent la situation des nouveaux arrivants dans la ville et parfois des personnes qui y vivent. Le deuxième exercice de cette pratique consiste à développer une solution pour aider le grand public à accéder plus facilement au transport de personnes. Il s'agit du transport public de masse transportant des passagers selon des itinéraires prédéterminés subdivisés en rues (taxi, autobus, véhicules particuliers, autobus, etc.). Ainsi, pour construire une interface utilisant une carte pour guider un utilisateur pour identifier l'endroit où aller compte tenu de l'endroit actuel.

Objectif : L'objectif principal de l'exercice pratique global est de permettre aux étudiants de comprendre comment analyser et concevoir des applications à l'aide de techniques orientées objet. D'autre part, ils apprendront à implémenter des logiciels en utilisant les techniques de programmation orientée objet. Ainsi, dans un premier temps, les élèves recueilleront des données sur certaines de leurs activités quotidiennes. Par la suite, ils utiliseront ces données pour concevoir et développer une application.

NB : L'ensemble des exercices est sur plus de 30. Cependant, les notes des étudiants seront maintenues sur 30. Pour les étudiants ayant plus de 30, le reste de leurs notes servira aux primes de contrôle continu.

Exercice 1 : collecte de données sur le parcours parcouru durant le semestre **(5pts)**
Le but de cet exercice est de collecter les données sur leur itinéraire tous les jours pendant ce semestre. Un itinéraire a un point de départ et un point d'arrivée et un coût. Parmi les informations pouvant être recueillies au cours de l'itinéraire, il y a la qualité de la route, les embouteillages, etc. Du point de départ au point d'arrivée, de nombreux itinéraires peuvent être parcourus. Chaque itinéraire correspond à une forme.

Ouvrez le formulaire (https://docs.google.com/forms/d/e/1FAIpQLSd72Y5bI5aVW0PxUOPZCXLZMxfPIZRexUU-SSP1jDzUdCIHvA/viewform?usp=sf_link) et remplissez-le pour chaque itinéraire.

Exercice 2 : collecte de données sur les aliments consommés au quotidien **(5pts)**
Le but de cet exercice est de collecter les données sur ce que vous mangez au quotidien. Le formulaire vous permet d'enregistrer tout ce que vous avez mangé dans la journée. Si vous avez mangé trois fois par jour, vous devez vous inscrire trois fois.

Ouvrir la formulaire (https://docs.google.com/forms/d/e/1FAIpQLSdnjccBdy4pIfejX0dpqdbPbtyEl105Unra9lENONexDQpDUg/viewf orm?usp=sf_link) et remplissez chaque fois que vous prenez un aliment.

Exercice 3 : mise en place de l'environnement de développement **(1pts)**

1. Configurez votre système d'exploitation (Linux)
2. Installez l'environnement de modélisation que vous prévoyez d'utiliser
3. Installez l'environnement de développement correspondant au langage de programmation que vous choisissez d'utiliser dans cet exercice
4. Écrivez le programme "Hello World" dans votre nouvel environnement de développement

Compte tenu du contexte et des données collectées, les étudiants doivent choisir une plate-forme à développer à l'aide de techniques de modélisation et de développement orientées objet.

Exercice 4 : Acteurs et cas d'utilisation **(2.5pts)**

Dans cet exercice, le langage UML est utilisé pour déterminer la limite du système.

1. Quels sont les acteurs ?
2. Donner les cas d'usage classés par acteurs (pour une bonne présentation, il faut les présenter dans le tableau).
3. Donner la description des cas d'utilisation en utilisant le formalisme textuel
4. Donnez la description des cas d'utilisation à l'aide de diagrammes de séquence
5. Donner les diagrammes de machine d'état des acteurs

Exercice 5ÿ: Diagrammes de classes et d'objets **(2,5ÿpts)**

1. Construire un tableau contenant les principales classes nécessaires à cet exercice **2.**

Proposer un diagramme de classes **3.** Proposer des diagrammes d'objets **4.** Compte tenu

des diagrammes de classes et d'objets, proposer un diagramme de packages

Exercice 6 : Pseudo-code **(2pts)**

Dans cet exercice, vous considérerez le formalisme OO

Proposer un pseudo code pour certaines fonctionnalités que vous pensez critiquer dans votre système

Exercice 5 : Design UI **(2pts)**

1. Concevoir l'interface utilisateur

principale **2.** Concevoir l'autre interface utilisateur

Exercice 6 : développement **(10pts)**

1. Utilisez les exercices 3 et 4 pour développer votre système
2. Utiliser l'exercice 2 pour construire le test fonctionnel
3. Testez votre programme

Exercice 7 : Développement de l'UI **(5pts)**

1. Utilisez l'exercice 5 pour développer l'interface

utilisateur **2.** Intégrez l'interface utilisateur à l'ensemble du système

3. Proposer la version mobile du système

Exercice 7 : (15pts)

Vous voulez construire une paire d'entiers positifs. Pour ce faire, on vous donne une liste de chiffres décimaux à utiliser. Vous devez utiliser chaque chiffre de la liste exactement une fois, mais vous pouvez choisir ceux à utiliser pour le premier entier et ceux à utiliser pour le deuxième entier. Vous pouvez également choisir l'ordre des chiffres dans chaque entier, sauf que vous ne pouvez pas mettre un zéro comme chiffre le plus significatif (le plus à gauche) dans l'un ou l'autre des entiers. Notez que vous ne pouvez pas non plus choisir un zéro pour un entier, car il ne serait pas positif.

Par exemple, on pourrait vous donner la liste [1,0,2,0,4,3]. Deux des paires valides que vous pouvez créer sont (200,143) et (3,12400). Les paires suivantes, en revanche, ne sont pas validesÿ:

• (0102,34) : a un zéro non significatif. •
(0,12340) : a un entier non positif. • (10,243) et
(12300,47)ÿ: la liste des chiffres de chacune de ces paires n'est pas exactement égale à la liste donnée de
chiffres.

Compte tenu de la liste des chiffres à utiliser, quelle est la différence absolue minimale entre les deux entiers construits qui peut être atteinteÿ?

Entrée

La première ligne de l'entrée donne le nombre de cas de test, T. Les lignes T suivent. Chaque ligne décrit un cas de test avec une seule chaîne de chiffres D. Chaque caractère de D est un chiffre que vous devez utiliser.

Sortie

Pour chaque cas de test, sortir une ligne contenant Case #xÿ: y, où x est le numéro du cas de test (à partir de 1) et y est le minimum possible. La différence absolue entre les deux entiers construits à partir de D selon les règles ci-dessus .

Limites

Limite de temps : 5 secondes.

Limite de mémoire : 1 Go.
1ÿTÿ100.

Chaque caractère de D est un chiffre décimal.
Au moins deux caractères de D ne sont pas 0.

Ensemble de test 1 (verdict visible)

2ÿ la longueur de Dÿ8.

Ensemble de test 2 (verdict visible)

2ÿla longueur de Dÿ36.

Échantillon

Sample Input

```
4
1234
0011
07080
0899
```

Sample Output

```
Case #1: 7
Case #2: 0
Case #3: 620
Case #4: 1
```

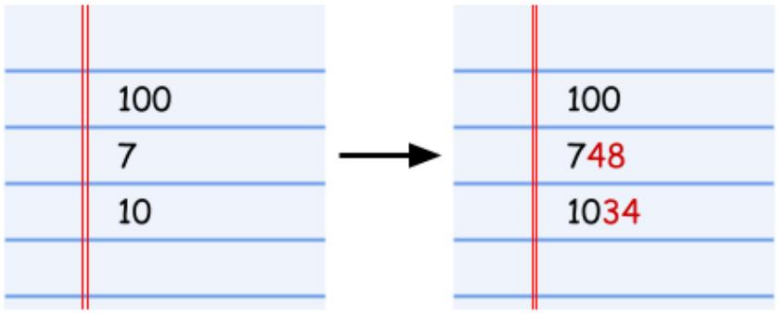
Exercice 8ÿ: 15pts

We have a list of integers $\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_N}$. We would like them to be in strictly increasing order, but unfortunately, we cannot reorder them. This means that usual sorting algorithms will not work.

Our only option is to change them by appending digits 0 through 9 to their right (in base 10). For example, if one of the integers is 10, you can turn it into $10\mathbf{0}$ or $10\mathbf{9}$ with a single append operation, or into $10\mathbf{34}$ with two operations (as seen in the image below).

Given the current list, what is the minimum number of single digit append operations that are necessary for the list to be in strictly increasing order?

For example, if the list is $100, 7, 10$, we can use 4 total operations to make it into a sorted list, as the following image shows.



Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ test cases follow. Each test case is described in two lines. The first line of a test case contains a single integer $\mathbf{N}$, the number of integers in the list. The second line contains $\mathbf{N}$ integers $\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_N}$, the members of the list.

Output

For each test case, output one line containing `Case #`x`:` y, where x is the test case number (starting from 1) and y is the minimum number of single digit append operations needed for the list to be in strictly increasing order.

Limits

.E

Time limit: 10 seconds.
Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$.

Test Set 1 (Visible Verdict)

$2 \leq \mathbf{N} \leq 3$.
$1 \leq \mathbf{X_i} \leq 100$, for all i.

Test Set 2 (Visible Verdict)

$2 \leq \mathbf{N} \leq 100$.
$1 \leq \mathbf{X_i} \leq 10^9$, for all i.

Sample

Sample Input

```
4
3
100 7 10
2
10 10
3
4 19 1
3
1 2 3
```

Sample Output

```
Case #1: 4
Case #2: 1
Case #3: 2
Case #4: 0
```

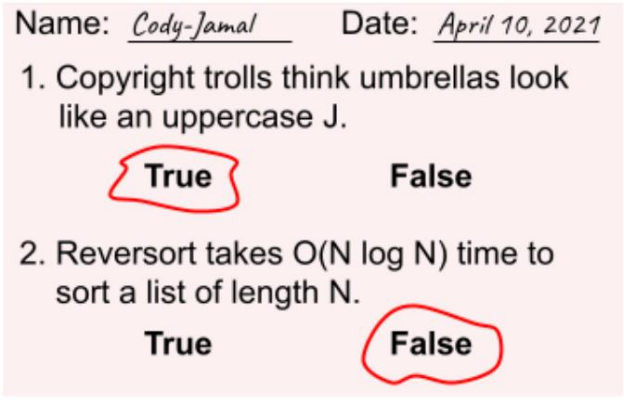
In Sample Case #1, the input is the same as in the example given in the problem statement. As the image shows, the list can be turned into a sorted list with 4 operations. Notice that the last two integers need to end up with at least 3 digits (requiring at least 3 append operations in total). If all of the final numbers had exactly three digits, the second would be larger than the third because it starts with a 7 instead of a 1. This means we cannot do it with fewer than 4 operations.

In Sample Case #2, notice that the list needs to be in strictly increasing order, so we have to do at least one operation. In this case, any valid append operation to the second integer works.

In Sample Case #3, we can use two append operations to get the list to $4, 19, \mathbf{193}$.

Exercice 9ÿ: 15pts

There is an exam with $\mathbf{Q}$ true or false questions. The correct answer to each question is either T or F. Each student taking the exam selects either T or F for each question, and the student's score is the number of questions they answer correctly.



There are $\mathbf{N}$ students who have already taken this exam. For each of those students, you know the answers they gave to each question and their final score. Assuming that any sequence of answers that is consistent with all of those students' scores has the same probability of being the correct sequence of answers, you want to maximize your own expected score. Determine what that expected score is and how to answer the questions so that you achieve it.

Input

The first line of the input gives the number of test cases, $\mathbf{T}$. $\mathbf{T}$ test cases follow. The first line of each test case contains two integers $\mathbf{N}$ and $\mathbf{Q}$: the number of students and the number of questions, respectively. Each of the next $\mathbf{N}$ lines contains a string $\mathbf{A_i}$ and an integer $\mathbf{S_i}$: the i-th student's answers and their score, respectively. The j-th character of $\mathbf{A_i}$ is either T or F, representing the answer the i-th student gave to the j-th question.

Output

For each test case, output one line containing `Case #`x`:` y z/w, where x is the test case number (starting from 1), y is a string representing a sequence of answers that yields the maximum expected score (in the same format as the input), and $\frac{z}{w}$ is the maximum expected score as an irreducible fraction (that is, w must be positive and of minimum possible value).

Time limit: 30 seconds.
Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 2021$.
The length of $\mathbf{A_i} = \mathbf{Q}$, for all i.
Each character of $\mathbf{A_i}$ is an uppercase T or an uppercase F, for all i.
$0 \leq \mathbf{S_i} \leq \mathbf{Q}$, for all i.
There exists at least one sequence of correct answers consistent with the input.

Test Set 1 (Visible Verdict)

$1 \leq \mathbf{N} \leq 2$.
$1 \leq \mathbf{Q} \leq 10$.

Test Set 2 (Hidden Verdict)

$1 \leq \mathbf{N} \leq 2$.
$1 \leq \mathbf{Q} \leq 40$.

Test Set 3 (Hidden Verdict)

$1 \leq \mathbf{N} \leq 3$.
$1 \leq \mathbf{Q} \leq 120$.

Sample

Note: there are additional samples that are not run on submissions down below.

Sample

Note: there are additional samples that are not run on submissions down below.

Sample Input

```
4
1 3
FFT 3
1 3
FFT 2
2 6
FFTTTF 2
FTFTFT 4
2 2
FF 1
TT 1
```

Sample Output

```
Case #1: FFT 3/1
Case #2: FFT 2/1
Case #3: FTFFFT 4/1
Case #4: TF 1/1
```

In Sample Case #1, given that the score for FFT is 3, the sequence of correct answers must be FFT.

In Sample Case #2, given that the score for FFT is 2, the sequence of correct answers is FFF, FTT, or TFT, each with probability $\frac{1}{3}$. Your best strategy is to answer FFT, to achieve the expected score of $\frac{1}{3} \times 2 + \frac{1}{3} \times 2 + \frac{1}{3} \times 2 = 2$.

In Sample Case #3, there are other answers that also achieve an expected score of 4, like FTFTFT.

In Sample Case #4, one of the questions' answer is T and the other one is F, but you do not know which is which. Answering TF or FT scores you 2 with probability $\frac{1}{2}$ and 0 with probability $\frac{1}{2}$, yielding an expected score of 1. Answering FF or TT guarantees a score of 1. Since any sequence of answers gives the same expected score, you can output any of them.

Additional Sample - Test Set 3

The following additional sample fits the limits of Test Set 3. It will not be run against your submitted solutions.

Sample Input

```
1
3 120
FFTFFFTFFFTTTTTTTFTFFFFFFTTTFTFFFTFTF
FFFTFFTTFFFFTFTFFTFFFTTTTTTFFFTTTFTTT
FFFTFTTFFFFFTFTFTTTTTTFFTTFTFFFTFFTT
```

Sample Output

```
Case #1: FFFTFTTTFFFFTFTFFTFTTTTTTTFF
```

In the Sample Case for Test Set 3, you can get an expected score over 65, which is higher than the actual score of any of the other students. Notice that both the numerator and denominator of the expected score can be significantly larger than 2^{64} (the numerator in this case actually exceeds 2^{97}).

Conseil : ne manquez pas les séances d'entraînement