MAPCOM SOLUTIONS INFORMATIQUES	**Marathon africain de codage 2020**	**30 avril** **C, C++, JAVA, Python**

anti-COVID-19

Problème A : Vélo

(ballon bleu)

La plupart des compteurs de vitesse pour vélo fonctionnent en utilisant un capteur à effet Hall fixé à la fourche avant du vélo. Un aimant est fixé à l'un des rayons de la roue avant de manière à ce qu'il s'aligne sur le capteur à effet Hall une fois par tour de roue. Le compteur de vitesse surveille le capteur pour compter les tours de roue. Si le diamètre de la roue est connu, la distance parcourue peut être facilement calculée si l'on sait combien de tours la roue a fait. En outre, si le temps nécessaire pour effectuer les tours est connu, la vitesse moyenne peut également être calculée. Pour ce problème, vous écrirez un programme pour déterminer la distance totale parcourue (en miles) et la vitesse moyenne (en miles par heure) en fonction du diamètre de la roue, du nombre de tours et de la durée totale du trajet. Vous pouvez supposer que la roue avant ne quitte jamais le sol, et qu'il n'y a pas de glissement ou de dérapage.

Entrée standard

La saisie consiste en plusieurs ensembles de données, un par ligne, du formulaire :

diamètre temps de rotation

Le diamètre est exprimé en pouces comme une valeur à virgule flottante. Le nombre de tours est une valeur entière. Le temps est exprimé en secondes comme une valeur à virgule flottante. L'entrée se termine lorsque la valeur de
Le nombre de révolutions est de 0 (zéro).

Sortie standard

Pour chaque ensemble de données, imprimer :

Voyage #N : distance MPH

Bien entendu, N doit être remplacé par le numéro de l'ensemble de données, la distance par la distance totale en miles (avec une précision de 2 décimales) et le MPH par la vitesse en miles par heure (avec une précision de 2 décimales). Votre programme ne doit pas générer de sortie pour le cas final lorsque le nombre de tours est égal à 0.

Marathon africain de codage 2020
anti-COVID-19

30 avril
C, C++, JAVA, Python

Constantes

Pour p, utilisez la valeur : 3.1415927.

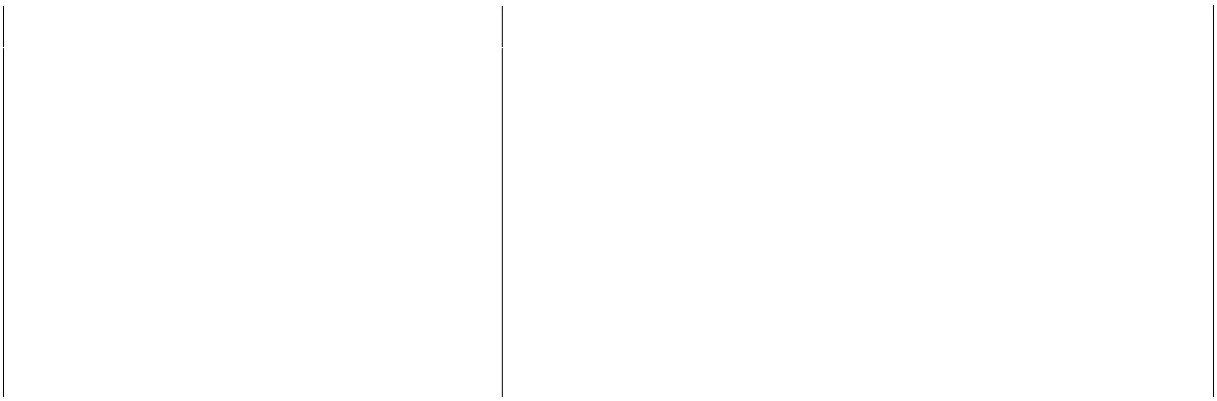
Il y a 5280 pieds dans un mile.

Il y a 12 pouces dans un pied.

Il y a 60 minutes dans une heure.

Il y a 60 secondes dans une minute.

Il y a 201,168 mètres dans un furlong.



Exemple d'entrée	Exemple de sortie
26 1000 5 27.25 873234 3000 26 0 1000	Voyage #1 : 1.29 928.20 Voyage #2 : 1179.86 1415.84

Problème B : les nombres binaires

(ballon rouge)

L'addition de nombres binaires est une tâche très simple, et très similaire à l'addition à la main de nombres décimaux. Comme pour les nombres décimaux, vous commencez par ajouter les bits (chiffres) une colonne à la fois, de droite à gauche. Contrairement à l'addition décimale, il y a peu de choses à mémoriser en ce qui concerne les règles d'addition des bits binaires :

```
0 + 0 = 0
1 + 0 = 1
0 + 1 = 1
1 + 1 = 10
1 + 1 + 1 = 11
```

Tout comme pour l'addition décimale, lorsque la somme dans une colonne est un nombre à deux bits (deux chiffres), le chiffre le moins significatif est écrit comme faisant partie de la somme totale et le chiffre le plus significatif est "reporté" dans la colonne de gauche suivante. Examinez les exemples suivants :

```
                    11  1  <-- Carry bits --> 1   11
  1001101             1001001                  1000111
+ 0010010           + 0011001                + 1010110
---------           ---------                ---------
  1011111             1100010                 10011101
```

Le problème d'addition à gauche n'a pas nécessité le report de bits, puisque la somme des bits dans chaque colonne était soit 1 ou 0, et non 10 ou 11. Dans les deux autres problèmes, il y avait bien des bits à transporter, mais le processus d'addition est encore assez simple.

Entrée standard

La première ligne d'entrée contient un entier N, (1 <= N <= 1000), qui est le nombre de problèmes d'addition binaire qui suivent. Chaque problème apparaît sur une seule ligne contenant deux valeurs binaires séparées par un seul caractère d'espacement. La longueur maximale de chaque valeur binaire est de 80 bits (chiffres binaires). Note : La longueur maximale peut être de 81 bits (chiffres binaires).

Sortie standard

Pour chaque problème d'addition binaire, imprimez le numéro du problème, un espace et le résultat binaire de l'addition. Les zéros de tête supplémentaires doivent être omis.

Exemple d'entrée	Exemple de sortie
3 1001101 10010 1001001 11001 1000111 1010110	1 1011111 2 1100010 3 10011101

Problème C : Changement (yellow ba lloon)

L'épicerie J.P. Flathead's Grocery engage de la main-d'œuvre bon marché pour tenir les caisses. Les personnes qu'il engage (généralement des lycéens) font souvent des erreurs en rendant la monnaie aux clients. Flathead, qui est un peu coincé, estime que ces erreurs lui font perdre plus d'argent qu'il n'en fait ; autrement dit, les employés ont tendance à rendre plus de monnaie aux clients qu'ils ne devraient en recevoir.

Flathead veut que vous écriviez un programme qui calcule le nombre de pièces de 25 cents (0,25 $), de 10 cents (0,10 $), de 5 cents (0,05 $) et de 10 cents (0,01 $) que le client devrait récupérer. Flathead veut toujours rendre la monnaie du client en pièces si le montant dû est de 5,00 $ ou moins. Il veut également rendre au client le plus petit nombre total de pièces. Par exemple, si le montant à rendre est de 1,24 $, le client devrait recevoir 4 pièces de 25 cents, 2 pièces de 10 cents, 0 pièce de 5 cents et 4 pièces de 10 cents.

Entrée standard

La première ligne d'entrée contient un nombre entier N qui est le nombre d'ensembles de données qui suivent. Chaque ensemble de données est constitué d'une seule ligne contenant un seul nombre entier qui est la variation due en cents, C, (1 ≤C ≤ 500).

Sortie standard

Pour chaque ensemble de données, imprimez le numéro de l'ensemble de données, un espace et la chaîne : Q QUARTER(S), D DIME(S), n NICKEL(S), P PENNY(S) Où Q est le nombre de quarts, D le nombre de pièces de dix cents, n le nombre de pièces de cinq cents et P le nombre de pièces de un cent.

Exemple d'entrée	Exemple de sortie
3 124 25 194	1 4 TRIMESTRE(S), 2 DIME(S), 0 NICKEL(S), 4 PENNY(S) 2 1 QUARTIER(S), 0 DIME(S), 0 NICKEL(S), 0 PENNY(S) 3 7 TRIMESTRE(S), 1 DIME(S), 1 NICKEL(S), 4 PENNY(S)

Marathon africain de codage 2020

anti-COVID-19

30 avril

C, C++, JAVA, Python

Problème D : La route la plus rapide vers Banikoara (ballon Oran ge)

Codjo : Je suis prêt pour le voyage à Banikoara

Bossi : Passons par Parakou

Assiba : Non, la route la plus rapide pour aller à Banikoara passe par Djougou

Vous êtes chargé de rédiger un programme qui, à partir d'une liste de villes et des distances séparant ces villes, donne la distance la plus courte à parcourir d'une ville A à une ville B.

Entrée standard

La première ligne d'entrée contient un seul entier P, (1 ≤ P ≤ 1000), qui est le nombre d'ensembles de données qui suivent. Chaque ensemble de données commence par une ligne contenant le nombre N des lignes restantes de l'ensemble de données (1 ≤ N ≤ 500), suivi d'un espace, suivi du nom d'une ville de départ, suivi d'un espace, suivi du nom d'une ville de destination. Chacune de ces N lignes contient le nom d'une ville de départ, suivi d'un espace, suivi du nom d'une ville de destination, suivi de la distance (en kilomètres) entre les deux villes. La distance sera un nombre entier et le nom de la ville sera une chaîne formée des caractères [a-z] [A-Z] et du signe "-".

Sortie standard

Pour chaque ensemble de données, vous devez générer une seule ligne de sortie contenant le nom d'une ville A et un espace, suivi du nom de la ville B, suivi d'un espace, suivi de la distance la plus courte à parcourir entre les villes A et B.

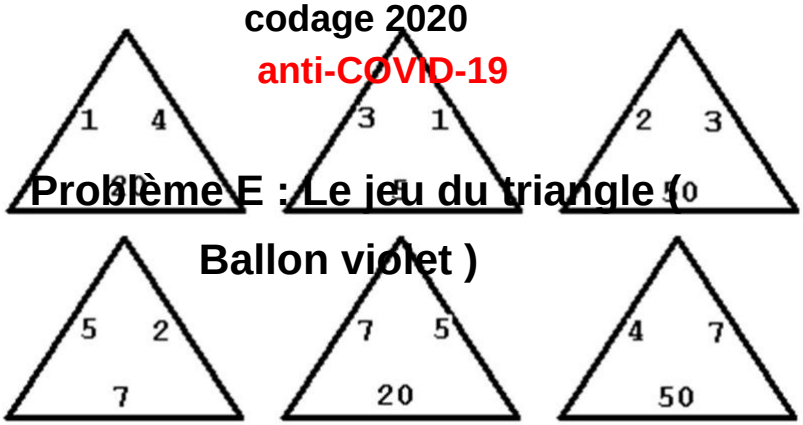
Exemple d'entrée	Exemple de sortie
1 4 Dassa-Zoume Banikoara Dassa-Zoume Djougou 270	Dassa-Zoume Banikoara 481

Marathon africain de codage 2020

anti-COVID-19

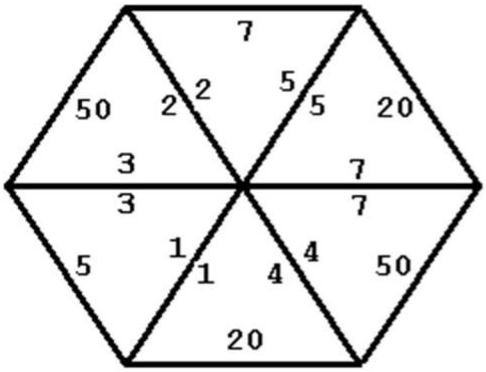
30 avril

C, C++, JAVA, Python

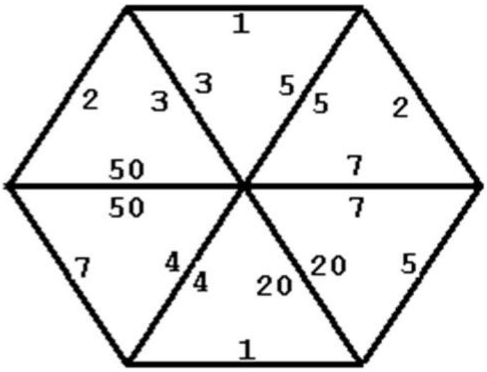


Problème E : Le jeu du triangle (Ballon violet)

Dans le jeu des triangles, vous commencez avec six triangles numérotés sur chaque bord, comme dans l'exemple ci-dessus. Vous pouvez faire glisser et tourner les triangles pour qu'ils forment un hexagone, mais l'hexagone n'est légal que si les bords communs à deux triangles portent le même numéro. Vous ne pouvez pas retourner un triangle. Deux hexagones légaux formés à partir des six triangles sont illustrés ci-dessous.



20+5+50+7+20+50 = 152



1+7+2+1+2+5 = 18

Le score d'un hexagone légal est la somme des chiffres des six bords extérieurs.

Votre problème est de trouver le score le plus élevé qui peut être obtenu avec six triangles particuliers.

Entrée standard

Le fichier d'entrée contient un ou plusieurs ensembles de données. Chaque ensemble de données est une séquence de six lignes avec trois nombres entiers de 1 à 100 séparés par des blancs sur chaque ligne. Chaque ligne contient les chiffres des triangles dans le sens des aiguilles d'une montre. Les ensembles de données sont séparés par une ligne ne contenant qu'un astérisque. Le dernier ensemble de données est suivi d'une ligne ne contenant qu'un signe de dollar.

Sortie standard

Pour chaque ensemble de données d'entrée, la sortie est une ligne contenant uniquement le mot "aucun" s'il n'y a pas d'hexagone légal ou le score le plus élevé s'il y a un hexagone légal.

Exemple d'entrée	Exemple de sortie
1 4 20 3 1 5 50 2 3 5 2 7 7 5 20 4 7 50 * 10 1 20 20 2 30 30 3 40 40 4 50 50 5 60 60 6 10 * 10 1 20 20 2 30 30 3 40 40 4 50 50 5 60 10 6 60	152 21 aucun