|  |
| --- |
| INPT |
| Les journées Nationales des Jeunes Développeurs |
| Problèmes pour le Real Time |
| 27/04/2014 |
|  |
|  |

|  |
| --- |
| Bonne chance ! |





## Agence

## a.(c|cpp|java)

Description du problème:

Le maroc vient de mettre en place la première agence de voyage inter-galactique. Chaque galaxie dispose d'une liste de taxes qu'il faut payer pour pouvoir y accéder. On représente la liste des taxes d'une galaxie par une chaine de N (1<= N <= 1000) bits. Les bits à '1' représentent les taxes qu'il faut payer pour accéder à la galaxie.

Une autre contrainte, c'est qu'un voyage entre deux galaxies n'est possible que si leurs chaines de bits ne diffère que dans une seule position.

Description de l'entrée:

Le fichier d'entrée contient plusieurs cas de test, chacun est décrit dans deux lignes. La première ligne contient N (le nombre de taxes dans l'espace), suivi par S (chaine de caractères de '0', et de '1', décrivant les taxes appliquées par la galaxie de départ), et E (chaine de caractères de '0', et de '1' décrivant les taxes de la galaxie de destination).

Le fichier d'entrée se termine par une ligne contenant 0 qu'il faut ignorer.

Description de la sortie:

Pour chaque cas dans le fichier d'entrée, afficher le coût total pour aller de la galaxie de départ vers la galaxie d'arrivée, selon le format ci-dessous.

|  |  |
| --- | --- |
| Exemple d'entrée | Exemple de sortie |
| 3 110 011  3 1 2  5 00000 11111  1 2 3 4 5  4 1111 1000  100 1 1 1  30 000000000000000000000000000000 111111111111111111111111111111  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30  0 | Case 1: 4  Case 2: 35  Case 3: 106  Case 4: 4960 |

## Balance

## b.(c|cpp|java)

Description du problème:

On dispose d'une balance, et de n (n>0) poids, respectivectivement : 1kg, 2kg, 4kg, ... 2^(n-1)kg.

En une séquence de N déplacements, on va mettre tous les poids dans la balance un à un. Calculer de nombre de manières de faire ces n déplacements, en évitant que le coté droit soit plus lourd que celui gauche.

On procède comme suit : Au début, on prend l'un des poids, et on le met dans le coté gauche de la balance. Ensuite, à chaque déplacement, on prend un poids, et on l'ajoute à la balance (soit au coté gauche, ou au coté droit) en maintenant le coté gauche plus lourd.

Description de l'entrée:

Le fichier d'entrée contient plusieurs cas de test, chacun est décrit par un entier n (1<=n<=100000) représentant le nombre de poids dont on dispose.

Le fichier d'entrée se termine par un cas où n==0 qu'il ne faut pas traiter.

Description de la sortie:

Pour chaque cas dans le fichier d'entrée, afficher le nombre de façons avec lesquelles on peut déplacer les n poids.

|  |  |
| --- | --- |
| Exemple d'entrée | Exemple de sortie |
| 1  2  0 | 1  3 |

## Concours

## c.(c|cpp|java)

Description du problème:

Il y a longtemps, dans une galaxie très lointaine, le général de l’armée a décidé d’organiser un concours pour sélectionner les leaders du future. Pour cela, il a rassemblé des enfants de différentes planètes dans l’un des vausseaux spatiales de l’académie de l’armée.

Le vaisseau spatial dispose de N étages numérotés de 1 à N, et dans chaque étage sont rassemblés les candidats d’une planète au nombre de Ck (k est l’indice de l'étage). Pour assurer le bon déroulement du concours, le générale libère un nombre du personnel du vaisseau pour surveiller les différents étages. Chaque surveillant peut au maximum contrôler M candidats d’un seul étage. Ceci dit, si le nombre de candidats d’un étage dépasse M, un nouveau surveillant est alors requis dans le même étage (2 surveillants pour M<Ck<=2M, et 3 surveillants pour 2M<Ck<=3M…).

Ne pouvant pas libérer autant de surveillants que requis, le général décide d’optimiser la distribution des candidats sur les étages. Il autorise alors le déplacement d’un ou plusieurs candidats de chaque étage k aux étages k-1 et k+1.

Description de l'entrée:

Le fichier d’entrée contient un ou plusieurs cas de tests. Chaque cas de test est représenté dans 2 lignes, la première ligne contient N entiers décrivant le nombre des candidats dans chaque étage (1<=N<=20 et 0<=Ck<=1000000000), et la deuxième ligne contient M le nombre max de candidat que peut surveiller un contrôleur (1<=M<=1000000000).

Description de la sortie:

Pour chaque cas de test, afficher dans le fichier de sortie un entier décrivant le nombre minimal de surveillants requis dans une ligne séparée.

|  |  |
| --- | --- |
| Exemple d'entrée | Exemple de sortie |
| 3 6 3  4  1 1 1 1  4  15 0 13 4 29 6 2  7 | 3  2  10 |

## Digits

## d.(c|cpp|java)

Description du problème:

Soit n>0 un entier. On désire calculer la somme de tous les nombres de n chiffres, puis la somme des chiffres du résultat.

Description de l'entrée:

Le fichier d'entrée contient plusieurs cas de test, chacun est décrit par un entier n (1<=n<=10000000000000) représentant le nombre de chiffres.

Le fichier d'entrée se termine par un cas où n==0 qu'il ne faut pas traiter.

Description de la sortie:

Pour chaque cas dans le fichier d'entrée, afficher la somme des chiffres de la somme des nombres de n chiffres.

|  |  |
| --- | --- |
| Exemple d'entrée | Exemple de sortie |
| 1  2  0 | 9  18 |

## Expression

## e.(c|cpp|java)

Description du problème:

Votre tâche est de décompresser une expression compressée sous l'une des formes suivante:

1- Une lettre en miniscule est un mot compressé.

2- *(e1 e2 ... et n)* où t et n sont deux entiers positifs, et les *ei* des mots compressés, est un mot compressé.

Bien sûr, un mot composé d'un seul caractère ne change pas après compression.

Pour décompresser le mot *(e1 e2 ... et n), on décompresse d'abord les* *ei*, on les concaténe pour former un mot, et on concaténe ce mot avec lui même n fois.Voici quelques exemples:

* x est décompressé en x,
* (t 3) devient après décompression: ttt,
* (a (b c 2) 3) devient: abcbcabcbcabcbc.

Description de l'entrée:

Le fichier d'entrée contient plusieurs cas de teste, chacun est décrit par une ligne qui se termine par le caractère '$'. La ligne peut contenir des espaces qu'il faut ignorer.

Le fichier d'entrée se termine par un cas qui contient uniquement le caractère '$', avec éventuellement des espaces avant, ou après. Ce cas n'est pas à traiter.

Description de la sortie:

Pour chaque cas dans le fichier d'entrée, afficher la ligne décompressée correspondante.

|  |  |
| --- | --- |
| Exemple d'entrée | Exemple de sortie |
| x$  (t 3)$  ( a ( b c 2 ) 3) $  $ | x  ttt  abcbcabcbcabcbc |

## Fox river

## f.(c|cpp|java)

Description du problème:

La prison de Fox River Chicago a connu une glorieuse fuite des frères Mickel Scofield et Lincoln Burrows mettant en cause tout le système de surveillance mis en place. Le directeur de la prison fait alors appel à une société pour mettre en place des caméras sur les différents couloirs de la prison. Ces caméras disposent de la fonctionnalité de reconnaissance faciale permettant alors de suivre le déplacement des agents de sécurité et des prisonniers. Ces caméras disposent chacune d’un champs de vision de 360°.

Toutefois, le prix d’achat et d’installation de ces caméras est considérable ! Le directeur de la prison fait alors un deuxième appel aux génies de l’algorithmique pour l’aider à minimiser le nombre d’installations requises.

Pour faciliter la tâche, la prison est schématisée par un plan sur 2 dimensions décrit par une grille contenant les caractères ‘#’ et ‘ .’, le caractère ‘#’ décrit un mur et le ‘.’ décrit un couloir. Tous les couloirs forment un arbre, il est alors possible d’accéder à un couloir à partir de n’importe quel autre couloir. Pour faciliter encore, les caméras sont supposées capables de contrôler les couloirs horizontalement et verticalement dans les 2 sens sans être capables de contrôler l’au-delà des murs.

Description de l'entrée:

Le fichier d’entrée contient un ou plusieurs cas de test. Chaque cas de test est décrit par un ensemble de lignes de même taille et qui décrivent la grille de la prison (la taille max de la grille est 50x50). Chaque cas de test est suivi par une ligne vide qui indique la fin de la grille.

Description de la sortie:

Pour chaque cas de test, imprimer sur le fichier de sortie un entier décrivant le nombre minimal de caméras requises dans une ligne séparée.

|  |  |
| --- | --- |
| Exemple d'entrée | Exemple de sortie |
| #...  ..##  #.##  .#.#.#.  .......  .#.#.#. | 2  4 |

## Game

## g.(c|cpp|java)

Description du problème:

Ali et Badr sont deux étudiants partageant la même chambre. En vue de preparer le dîner,ils décident de jouer un jeu simple, le perdant aura bien sûr la tâche de le préparer

Le jeu consiste en une ligne de pièces de monnaies avec des valeurs Vi (1=<i<=n), chaque joueur est autorisé à prendre une seule pièce de la tête ou la queue de la ligne, puis l’autre joueur en fait ainsi. A la fin (la ligne ne contienne plus de pièces), le joueur qui a collecté la plus grande somme gagne. Nous supposerons que les deux joueurs sont aussi intelligents pour faire les meilleurs choix

Description de l'entrée:

Le fichier d’entrée contient plusieurs lignes , chaque ligne contient un entier 1<=n<=100, suivi de n entiers Vi: les valeurs des pièces de monnaies

Le fichier d’entrée se termine lorsque n=0 qu’il ne faut pas traiter

Description de la sortie:

Pour chaque ligne d’entrée, afficher la somme capitalisée par le gagant

|  |  |
| --- | --- |
| Exemple d'entrée | Exemple de sortie |
| 4 5 3 7 10  0 | 15 |

## Hasard

## h.(c|cpp|java)

Description du problème:

Oscar’s Oasis est une série télévisée de dessin animés très populaire. Dans un épisode, les 3 amis Popy, Harchi et Buck jouent contre Oscar, le prix du jeu est une caisse de pommes tombée par la camionnette de Roco.

Le jeu consiste à mettre R balles rouges et B balles bleues dans un panier et jouer à tour de rôle suivant les contraintes suivantes :

* Avant de jouer, M balles sont retirées aléatoirement du panier (en aucun cas, les balles rouges sont toutes retirées)
* Oscar joue en premier
* A chaque tour, le joueur retirer 1, 2 ou 3 balles puis annonce le nombre de balle de chaque couleur retiré
* Le joueur qui retire la dernière balle rouge perd le jeu

Votre mission et de trouver la probabilité qu’Oscar gagne le défi sachant que les 2 parties cherchent à optimiser leur chance de gagner.

Malheureusement pour Oscar, même s’il gagne le jeu il ne reçoit pas le prix promis 

Description de l'entrée:

Le fichier d’entrée contient un ou plusieurs cas de test. Chaque cas de test est représenté par une ligne contenant 3 entier R, B et M (1<=R,B<=100 et 0<=M<R).

Description de la sortie:

Pour chaque cas de test, imprimer sur le fichier de sortie sur une ligne séparée la probabilité que Oscar gagne. La probabilité doit être à la précision de 1e-9.

|  |  |
| --- | --- |
| Exemple d'entrée | Exemple de sortie |
| 1 1 0  1 2 0  2 1 0  2 2 1 | 0.5  0.333333333  0.666666667  0.5 |