Questionnaire de l'Epreuve de Sélection Préliminaire
(sur 100 points)

Problème I – 3N+1 — 10 points

Considérons la suite numérique suivante. Le premier terme a_1 de la suite est égal à un nombre entier positif strictement supérieur à 1. Le terme suivant est calculé de la façon suivante:

$a_i = \frac{a_{i-1}}{2}$ si a_{i-1} est pair

$a_i = 3a_{i-1} + 1$ si a_{i-1} est impair

A première vue, on pourrait penser que cette suite ne converge pas parce que, pour des termes impairs, on multiplie par 3 et pour des termes pairs, on divise par 2. En conséquence, les nombres devraient devenir de plus en plus grands. Or, cette suite semble converger vers 1, ce qui n'a pas encore été démontré jusqu'aujourd'hui!

Problème

Ecrivez un programme pour calculer les termes de cette suite.

Entrée

Le programme accepte au clavier un nombre entier positif strictement supérieur à 1 et inférieur ou égal à 1000000.

Sortie

Le programme affiche la suite à l'écran.

Exemple

Entrée: 11
Sortie: 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Remettez le programme sous le nom SUITE.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable SUITE.EXE correspondant au programme.

Problème II – La tirelire d'Alexandre — 20 points

A l'occasion du quatrième anniversaire du petit Alexandre, son père lui avait fait cadeau d'une tirelire avec un cadenas numérique ayant une combinaison de 3 chiffres. Le gamin avait tout de suite défini une combinaison de chiffres pour sécuriser sa tirelire. Dès lors, Alexandre y introduit régulièrement des pièces de monnaie.

A l'âge de 6 ans, la tirelire du petit Alexandre est pleine et il décide de l'ouvrir. Mais hélas, il ne se rappelle plus la combinaison de chiffres pour ouvrir le cadenas. La seule chose qu'il se rappelle est que la combinaison ne comporte pas de chiffre zéro et qu'aucun chiffre n'y figure plus d'une fois.

Problème

Ecrivez un programme qui aide le petit Alexandre à trouver toutes les combinaisons possibles ainsi que le nombre de combinaisons possibles.

Sortie

Le programme écrit les combinaisons trouvées dans le fichier TIRELIRE.TXT, un nombre par ligne. Les combinaisons doivent être triées de manière croissante. A la fin, donc après la dernière combinaison, le nombre de combinaisons est indiqué dans une ligne supplémentaire.

Remettez le programme sous le nom TIRELIRE.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable TIRELIRE.EXE correspondant au programme.

Problème III – Money — 30 points

Dans votre portefeuille, se trouvent différentes coupures de billets en euros. Quels montants pouvez-vous dépenser exactement, sans qu'on doive vous rendre de la monnaie?

Problème

Ecrivez un programme qui calcule tous les montants possibles.

Exemple

Si votre portefeuille contient 1 billet de 5 et 1 billet de 20, vous pouvez dépenser exactement 5, 20 ou 25 euros.

Entrée

Le fichier d'entrée, MONEYIN.TXT, contient 7 nombres entiers, séparés par un espace, représentant le nombre de billets de 5, 10, 20, 50, 100, 200 et 500 euros.

Pour l'exemple, le fichier d'entrée est: 1 0 1 0 0 0 0

Sortie

Le fichier de sortie, MONEYOUT.TXT, contient tous les montants que vous pouvez générer à l'aide des billets donnés, un par ligne. Les montants doivent être triés de manière croissante. Le fichier ne doit pas contenir des répétitions.

Pour l'exemple, le fichier de sortie est:
5
20
25

Restriction

Votre portefeuille ne contiendra pas plus de 10 billets de chaque sorte.

Remettez le programme sous le nom MONEY.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable MONEY.EXE correspondant au programme.

Problème IV – Sudoku — 40 points

Sudoku est un jeu de réflexion très à la mode pour l'instant. On peut trouver toutes les informations nécessaires sur Internet à l'adresse **www.sudoku.com**. Dans ce problème, nous voulons développer un programme aidant à la résolution en appliquant la stratégie des "solitaires nus". Cette stratégie utilise ce qu'on appelle la "liste des candidats".

Etudions un exemple. Soit le Sudoku ci-contre. La liste des candidats est établie en examinant toutes les cases vides et en y inscrivant, à l'aide de "petits chiffres", tous les candidats possibles. Ainsi, dans la première case en haut à gauche, les candidats possibles sont le 1, le 3 et le 9. Les chiffres 2, 4, 5, 6, 7 et 8 se trouvant déjà dans la première ligne, la première colonne respectivement le premier sous-carré 3x3 ne peuvent plus être mis dans la première case.

	7	6	2	4	8		5	
			7	5				
	8					6		
							7	4
4		8		1		3		2
2	3							
		3					6	
				2	9			
	9		8	6	5	1	4	

La figure I montre la liste des candidats pour toutes les cases du Sudoku. On remarque par exemple que la dernière case de la première colonne ne contient plus qu'un seul candidat, le 7, qu'on appelle alors le "solitaire nu".

Vu qu'aucun autre chiffre n'y est possible, ce 7 doit être placé dans cette case. Après le placement du 7, la liste des candidats doit être mise à jour. Tous les candidats 7 se trouvant dans la même ligne, la même colonne ou le sous-carré 3x3 qui contient ce solitaire nu doivent être enlevés. Ce qui donne la figure II.

Comme on vient d'enlever le candidat 7 dans toute la dernière ligne, un nouveau solitaire nu apparaît dans la troisième colonne, le 2, qu'on peut donc placer maintenant. Ensuite, on enlève tous les candidats 2 dans cette dernière ligne, dans la troisième colonne et dans le sous-carré 3x3 qui contient ce solitaire nu. On répète ce procédé tant qu'il reste un solitaire nu. On peut ainsi éventuellement arriver à un Sudoku résolu.

Figure I

Figure II

Problème

Ecrivez un programme qui accepte un Sudoku donné dans le fichier d'entrée et qui recherche et place les solitaires nus. A la fin du programme, s'il ne reste plus de solitaires nus, le Sudoku obtenu est écrit dans le fichier de sortie.

Entrée

Le Sudoku donné se trouve dans le fichier d'entrée "SUDOIN.TXT"; les cases vides sont marquées par le 0 (zéro).

Pour l'exemple, le fichier d'entrée est:

```
0 7 6 2 4 8 0 5 0
0 0 0 7 5 0 0 0 0
0 8 0 0 0 0 6 0 0
0 0 0 0 0 0 0 7 4
4 0 8 0 1 0 3 0 2
2 3 0 0 0 0 0 0 0
0 0 3 0 0 0 0 6 0
0 0 0 0 2 9 0 0 0
0 9 0 8 6 5 1 4 0
```

Sortie

Le Sudoku résultant est écrit dans le fichier de sortie "SUDOOUT.TXT"; les cases vides sont marquées par le 0 (zéro).

Pour l'exemple, le fichier de sortie est:

```
3 7 6 2 4 8 9 5 1
0 0 0 7 5 0 4 0 8
0 8 0 0 0 0 6 0 7
0 0 0 0 0 0 0 7 4
4 0 8 0 1 0 3 9 2
2 3 0 0 0 0 0 1 6
0 0 3 0 7 0 2 6 9
0 0 0 0 2 9 7 8 5
7 9 2 8 6 5 1 4 3
```

Remettez le programme sous le nom SUDOKU.xxx, avec xxx=PAS ou C(PP). Remettez également le fichier binaire exécutable SUDOKU.EXE correspondant au programme.