

Durée : 1h30

Nom chinois :

Numéro étudiant :

Classe :

► 1. Écrivez la fonction `sommeDesDiviseurs` qui renvoie la somme des diviseurs stricts d'un nombre entier positif `n` (> 0) passé en paramètre. Par exemple, la somme des diviseurs stricts de 78 est égale à 90. Notez que 78 n'est pas diviseur strict de lui-même.

[illegible]

- 2. Un nombre est dit parfait s'il est égal à la somme de ses diviseurs stricts. Par exemple, $28 = 1 + 2 + 4 + 7 + 14$. Écrivez la fonction booléenne `estParfait` qui teste si son paramètre (> 0) est un nombre parfait ou non.

- 3. Écrivez la fonction `main` qui lit sur l'entrée standard un entier `n` (>0) et qui écrit sur la sortie standard tous les nombres parfaits sur l'intervalle $[1; n]$.

Question 2. Nombres binaires

On représente un entier long non signé (**unsigned long int**) sous forme binaire à l'aide d'un tableau d'entiers courts (**short**) contenant des 0 et des 1. Le bit de poids faible (2^0) est à l'indice 0, et le bit de poids fort ($2^{\text{NB_BITS}-1}$) est à l'indice `NB_BITS-1`. Par exemple, si `NB_BITS` est égal à 8, le tableau qui contient `{0, 1, 1, 0, 1, 0, 0, 0}` représente l'entier décimal non signé 22. On définira le type `binaire`, comme suit :

```
typedef short binaire[NB_BITS];
```

- 4. À l'aide d'un **define**, définissez la constante `NB_BITS` égale au nombre d'éléments du tableau nécessaires à la représentation d'un entier long non signé (**unsigned long int**).

- 5. Écrivez la fonction `binaireToLongInt` qui calcule et renvoie la valeur décimale d'un nombre binaire. Vous ne ferez aucune évaluation à la puissance. Cette fonction possède l'en-tête suivant :

```
unsigned int binaireToLongInt(const binaire b) {
```

- 6. Écrivez la procédure `et` qui renvoie la conjonction de deux nombres binaires représentés par des tableaux de même taille.

Question 3. Matrices

- 7. Écrivez la procédure `printSubMat` qui écrit sur la sortie standard une sous-matrice issue d'une matrice de réels **double**. Cette procédure possède 7 paramètres :
- une matrice `mat`, un tableau de réels **double** à 2 dimensions ;
 - 2 entiers `m` et `n` qui correspondent aux nombres de lignes et de colonnes ;
 - 4 entiers `i1`, `i2`, `j1` et `j2`, avec $i1 \geq 0$ et $i1 < m$ et $i2 \geq 0$ et $i2 < m$ et $j1 \geq 0$ et $j1 < n$ et $j2 \geq 0$ et $j2 < n$ et $i1 \leq i2$ et $j1 \leq j2$ (vous vérifierez la validité des indices) ;
- La procédure `printSubMat` écrit sur la sortie standard la sous-matrice de `mat` formée des lignes `i1` à `i2` et des colonnes `j1` à `j2`.

- 8. Soit une matrice $m \times n$ qui contient des entiers (**int**). Écrivez la fonction booléenne **creuse** qui renvoie *vrai* si plus de la moitié des entiers contenus dans la matrice sont égaux à 0, et *faux* sinon. Cette fonction possède l'en-tête suivant :

```
int creuse(const int m, const int n, const double mat[][n])
```

[illegible]