Exercice 1 – Fractions

Cet exercice consiste à manipuler des nombres fractionnaires sans passer par des valeurs décimales.

1. Définir un type structuré **fraction** permettant de représenter une fraction.
2. Écrire une procédure **saisie** qui permette de saisir une fraction au clavier (sous forme de deux entiers).
3. Écrire une procédure **affichage** qui affiche une fraction (ex: '2/5').
4. Écrire une fonction **mult** qui multiplie deux fractions (sans simplifier).
5. Écrire une fonction **add** qui somme deux fractions (sans simplifier).
6. Écrire les fonctions **opp** et **inv**, qui renvoient respectivement l'opposé et l'inverse de la fraction passée en paramètre.
7. Écrire les fonctions **soustr** et **div**, qui réalisent respectivement la soustraction et la division de deux fractions.
8. Écrire une fonction **pgcd** qui calcule le plus grand commun diviseur de deux entiers.
9. En utilisant la fonction pgcd, écrire une fonction **simpl** qui simplifie la fonction passée en paramètre en la rendant irréductible, et en imposant un dénominateur positif.
10. Modifier **mult**, **add**, **soustr** et **div** afin qu'ils renvoient une fraction simplifiée.

Soit le type suivant représentant un nombre mixte :

```
struct nbmixte {
        int signe ;
        int ent ;
        fraction frac ;
} ;
```

Un nombre mixte est un nombre fractionnaire dont l'écriture associe un entier et une fraction dont le numérateur est nécessairement inférieur au dénominateur. Ce nombre vaut la somme de l'entier et la fraction (par exemple, 2 1/3 pour 7/3). Le signe vaut pour l'ensemble du nombre, et pas seulement l'entier (par exemple, - 5 3/5 pour -28/5). La valeur du signe sera 1 pour les nombres positifs, et -1 pour les nombres négatifs.

11. Écrire une fonction **mixte** qui traduit une fraction en un nombre mixte.
12. Écrire une fonction **addM** qui somme deux nombres mixtes.
    a) en convertissant les nombres mixtes en fractions, et réciproquement ;
    b) directement à partir de la représentation en nombres mixtes [*difficile*].

Questions supplémentaires (rappels sur les chaines de caractères)

Dans la suite, on n'utilisera pas le type nbmixte.

13. Écrire les sous-programmes **cherche** et **extrait** permettant respectivement de déterminer la position d'un caractère dans une chaine (-1 si celui-ci n'apparait pas), et de renvoyer une sous-chaine extraite d'une autre (à partir d'une position et d'une longueur).

14. Écrire un sous-programme **convertit** permettant de convertir en nombre entier une chaine de caractères représentant un nombre.

15. Écrire un sous-programme **valide** qui détermine si une chaine de caractères décrit un nombre fractionnaire valide (par exemple : 3624, -23/3, 36/164, 2/-3, 0/45, mais pas /36, 2-3/4, 5/0, 7-9, 8a).

16. En utilisant les sous-programmes précédents, écrire une procédure **saisierapide** qui permette de saisir une fraction au clavier, directement dans le format d'affichage (par exemple, 3/7).

17. Écrire un programme principal où l'utilisateur saisit deux fractions, puis affiche leur somme, produit, différence et rapport.

Exercice 2 – Ensembles de points

On désire représenter des points dans le plan. Chaque point est codé sous la forme d'un enregistrement contenant une coordonnée *x* et une coordonnée *y*.

1. Définir un type structuré **point** permettant de représenter un point.
2. Écrire une fonction déterminant si deux points sont **confondus**.
3. Écrire une fonction qui calcule la **distance** euclidienne entre deux points.
4. Écrire une fonction qui calcule le **milieu** d'un segment constitué de deux points.

On désire maintenant représenter des ensembles de points. Pour cela on stockera des points dans un tableau.

5. Définir un type **enspoints** représentant un ensemble de points sous forme d'un enregistrement composé d'un tableau et de sa taille effective.
6. Écrire une procédure de **saisie** d'un ensemble de points.
7. Écrire une fonction calculant le **centre** de gravité d'un ensemble de points.
8. Écrire une procédure **reduit** permettant de supprimer les éventuels doublons contenus dans un tableau de type enspoints, tout en conservant l'ordre dans lequel ils sont stockés.
9. Écrire une fonction **plusproche** qui retourne, parmi un ensemble de points passé en paramètre, le point le plus proche de l'origine (0,0).
10. Écrire une fonction **distmin** qui calcule la plus petite distance entre deux points d'un ensemble de points (non nécessairement consécutifs).
11. Écrire une fonction **tousalignes** qui détermine si un ensemble de points est composé de points tous alignés.
12. Écrire une fonction **contours** qui calcule l'ensemble des sommets du plus petit rectangle dont les côtés sont parallèles aux axes d'un repère orthonormé, et permettant de contenir tous les points d'un ensemble passé en paramètres. La fonction retournera dont un ensemble de 4 points.
13. Écrire une procédure qui **trie** les points par abscisse croissante.
14. Écrire une fonction qui détermine l'**image** d'un réel x par la fonction affine par morceaux continue représentée par un ensemble de points.