

Examen de Examen de Langage C

Durée : 1h30

Aucun document autorisé – Mobiles interdits

Notez que les affirmations (antécédents, conséquents, rôles, et invariants) dans vos codes C entreront pour partie dans la note finale.

1 Sorte de caractères

- 1. Déclarez le type énuméré `SorteCar` qui définit 4 catégories de caractères : les lettres alphabétiques, les chiffres, les espaces et les autres (le complémentaire). Les valeurs de ce type seront LETTRE, CHIFFRE, ESPACE et AUTRE.

```
enum SorteCar { LETTRE, CHIFFRE, ESPACE, AUTRE} ;
```

- 2. Écrivez la fonction `typeDeCar` qui prend en paramètre un caractère et qui renvoie la catégorie du caractère (de type `SorteCar` précédemment défini). Vous pouvez utiliser les fonctions de `ctype.h`.

```
/*
 * Antécédent : c un caractère quelconque
 * Rôle : renvoie la catégorie du caractère c
 */
enum SorteCar typeDeCar(const char c) {
    if (isalpha(c)) return LETTRE;
    if (isdigit(c)) return CHIFFRE;
    if (isspace(c)) return ESPACE;
    return AUTRE;
}
```

2 Tableaux

- 3. Écrivez la fonction `somme` qui calcule la somme des éléments d'un tableau de n entiers (`int`). L'en-tête de cette fonction est :

```
int somme(const int n, const int t[]) {
```

```
/*
 * Antécédent : n>0
 * Rôle : renvoie la somme des n valeurs de t
 */
int somme(const int n, const int t[]) {
    int s=0;
    for (int i=0; i<n; i++)
        s+=t[i]; // s =  $\sum_{k=0}^i t[k]$ 
    // s =  $\sum_{k=0}^{n-1} t[k]$ 
    return s;
}
```

- 4. Écrivez la fonction `ligneMin` qui renvoie l'indice de la ligne d'une matrice $m \times n$ d'entiers (`int`) dont la somme de ses valeurs est la plus petite. Pensez à utiliser la fonction `somme` de l'exercice précédent.

```
/*
 * Antécédent : m>0 et n>0
 * Rôle : renvoie l'indice de la ligne dans l matrice mat
 *          dont la somme de ses valeurs est la plus petite
 */
int ligneMin(const int m, const int n, const int mat[][n]) {
    int imin, min=INT_MAX;
    for (int i=0; i<m; i++) {
        // calculer la somme des entiers de la ligne i
        int si = somme(n, mat[i]);
        // est-elle la plus petite ?
        if (si<min) {
            min = si;
            // mémoriser l'indice de la ligne
            imin = i;
        }
    }
    //
    return imin;
}
```

3 Plan cartésien

On représente les coordonnées d'un point p du plan cartésien par deux réels (**double**) x et y tels que $p = (x, y)$.

- 5. Écrivez en C la fonction `pente` qui renvoie la pente d'une droite qui passe par 2 points distincts de coordonnées (x_1, y_1) et (x_2, y_2) . On rappelle que cette pente est égale à $\frac{y_2 - y_1}{x_2 - x_1}$.

```
/*
```

```

* Antécédent :  $x1 \neq x2$ ;
* Rôle : renvoie la pente de la droite qui passe par les 2 points
*         distincts de coordonnées (x1,y1) (x2,y2)
*/
double pente(const double x1, const double y1,
             const double x2, const double y2)
{
    assert(x1!=x2);
    // calculer et renvoyer la pente
    return (y2-y1)/(x2-x1);
}

```

```

    printf("y = %.2lf\n", Ay);
else {
    // droite quelconque  $\Rightarrow$  calculer la pente a
    double a = pente(Ax, Ay, Bx, By);
    //  $b = y - ax \Rightarrow$  calculer b à partir de A(Ax, Ay)
    double b = Ay - a * Ax;
    // écrire l'équation de la droite sur la sortie standard
    printf("y = %.2f x %c %.2f\n", a, (b > 0 ? '+' : '-'), fabs(b));
}
return EXIT_SUCCESS;
}

```

► 6. Écrivez en C la fonction main qui :

1. déclare les variables Ax, Ay, Bx, By de type **double** qui représentent les coordonnées de 2 points A et B;
2. lit sur l'entrée standard 4 réels (**double**) et les affecte aux variables Ax, Ay, Bx, By;
3. calcule la pente a de la droite qui passe par A et B;
4. écrit sur la sortie standard l'équation $y = ax + b$ de la droite qui passe par A et B;

Par exemple, l'exécution de cette fonction main pourrait produire :

```

A ? 3 -8
B ? 5 10
l'équation de la droite passant par A et B est : y = 9.00x - 35.00

```

```

int main(void) {
    double Ax, Ay, Bx, By;

    // lire les coordonnées du point A
    printf("A(x,y) = ");
    scanf("(%lf,%lf)", &Ax, &Ay);
    // lire les coordonnées du point B
    printf("B(x,y) = ");
    scanf("(%lf,%lf)", &Bx, &By);

    // écrire sur la S/S standard la distance entre les points A et B
    printf("distance(A,B) = %lf\n", distance(Ax, Ay, Bx, By));

    // déterminer la droite qui passe par A et B
    if (Ax==Bx && Ay==By) {
        fprintf(stderr, "2 points confondus, pas de droite\n");
        return EXIT_FAILURE;
    }
    // les points sont distincts => trouver l'équation de droite
    printf("l'équation de la droite passant par A et B est : ");
    if (Ax==Bx)
        // droite verticale  $\Rightarrow a=1$  et  $b=Ax$ 
        printf("x = %.2lf\n", Ax);
    else
        // Ax!=Bx
        if (Ay==By)
            // droite horizontale  $\Rightarrow a=0$  et  $b=Ay$ 

```