

Examen de Info C

Durée : 1h
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 Voyelles

- 1. Écrivez la fonction `estUneVoyelle` qui prend en paramètre un caractère et qui renvoie 1 (*vrai*) si ce caractère est une voyelle et 0 (*faux*) sinon. On considérera les 6 voyelles minuscules *a,e,i,o,u,y* et les 6 voyelles majuscules *A,E,I,O,U,Y*.

```
/*
 * Rôle : renvoie 1 si c est une voyelle et 0 sinon.
 */
int estUneVoyelle(const char c) {
    switch (tolower(c)) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
        case 'y': return 1;
    }
    // c n'est pas une voyelle
    return 0;
}
```

- 2. Écrivez la fonction `compterVoyelles` qui prend en paramètre un tableau de caractères `t` et qui renvoie le nombre de voyelles qu'il contient. Le nombre de caractères `n` contenus dans le tableau `t` et le tableau `t` sont passés en paramètres de la fonction.

```
/*
 * Antécédent : n>0
 * Rôle : renvoie le nombre de voyelles contenues dans t
 */
int compterVoyelles(const int n, const char t[]) {
    int nbVoyelles=0;
    for (int i=0; i<n; i++)
        if (estUneVoyelle(t[i]))
            // le caractère courant est une voyelle
```

```
        nbVoyelles++;
        // on a compté toutes les voyelles
    return nbVoyelles;
}
```

- 3. En utilisant la fonction `compterVoyelles` précédente, écrivez la fonction `main` qui écrit sur la sortie standard le nombre de voyelles contenues dans le mot *Anticonstitutionnellement*.

```
int main(void) {
    printf("nb voyelles : %d\n", compterVoyelles(25, "Anticonstitutionnellement"));
    return EXIT_SUCCESS;
}
```

2 Matrices

- 4. Écrivez la procédure `initAléa` qui initialise de façon aléatoire une matrice `mat` d'entiers (`int`) de dimension $m \times n$. Vous utiliserez la fonction `rand`.

```
/*
 * Antécédent : m>0 et n>0
 * Rôle : initialise de façon aléatoire la matrice mat(m,n)
 */
void initAléa(const int m, const int n, int mat[][n]) {
    for (int i=0; i<m; i++)
        // initialiser la ième ligne
        for (int j=0; j<n; j++)
            mat[i][j] = rand();
}
```

- 5. Écrivez la fonction booléenne `auMoinsNzéros` qui renvoie 1 (*vrai*), si la demi-matrice supérieure d'une matrice carrée $n \times n$ d'entiers contient *au moins* k (> 0) zéros, sinon elle renvoie 0 (*faux*). La matrice, sa dimension et le nombre k de zéros sont passés en paramètres.

```
/*
 * Antécédent : k>0
 * Rôle : renvoie 1 (vrai) si la demi-matrice supérieure de mat n×n
 *         contient au moins k zéros, et 0 (faux) sinon
 */
int auMoinsNzéros(const int n, const int mat[][n], const int k) {
    int nbZéros = k;
    // parcourir la demi-matrice supérieure de mat
    // diagonale incluse
    for (int i=0; i<n; i++) {
```

```

    for (int j=i; j<n; j++)
        if (mat[i][j]==0)
            if (--nbZéros==0)
                // il y a bien au moins k zéros dans la demi-matrice sup
                return 1;
    }
    // il y a moins de k zéros dans la demi-matrice sup de mat
    return 0;
}

```

3 Matrices et Vecteurs

- 6. Écrivez la procédure `vecteurSomme` qui prend en paramètre une matrice `mat` de dimension $m \times n$ et un vecteur `vect` de dimension m . Cette procédure affecte à chaque `vect[i]` la somme des valeurs entières de chaque ligne i de la matrice. *Structurez correctement le code de votre réponse.*

```

/*
 * antécédent : n>0
 * rôle : renvoie la somme des entiers
 *         du tableau t
 */
int somme(const int n, const int t[]) {
    int som=0;
    for (int i=0; i<n; i++)
        som+=t[i];
    // som =  $\sum_{i=0}^n t[i]$ 
    return som;
}

/*
 * antécédent : m>0 et n>0 et mat initialisée
 * rôle : crée le vecteur v formé de la somme
 *         des entiers de chacune des lignes
 *         de la matrice mat
 */
void VecteurSomme(const int m, const int n,
                  const int mat[][n], int vect[])
{
    for (int i=0; i<m; i++)
        // calculer la somme des entiers
        // de la ligne : mat[i]
        vect[i] = somme(n, mat[i]);
}

```