

Examen de Info C

Durée : 1h30

Aucun document autorisé

Mobiles interdits

Nom :

Prénom :

- 1. En utilisant uniquement la notation de pointeur et à l'exclusion de toute autre fonction (sauf `isdigit`), écrivez en C la fonction `sansLesChiffres` qui prend en paramètre deux chaînes de caractères `s1` et `s2`. Cette fonction copie `s2` dans `s1` en enlevant tous les chiffres présents dans `s2`. De plus, elle renvoie le nombre de chiffres supprimés. Par exemple, l'appel suivant : `sansLesChiffres(s, "jeudi 14 juin 2018")` renvoie la valeur 6 et `s` est égale à `"jeudi juin"`.

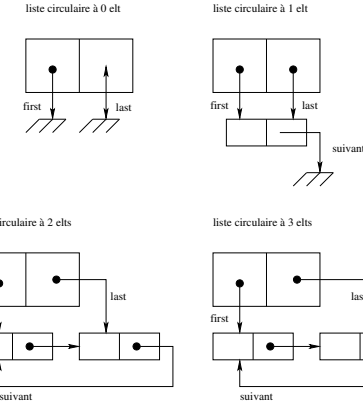
```
/*
 * Rôle : affecte à s1 la chaîne s2 dans laquelle les
 * chiffres ont été supprimés. De plus, elle renvoie le
 * nombre de chiffres supprimés.
 */
int sansLesChiffres(char *s1, const char *s2) {
    int n=0;
    while (*s2) {
        if (!isdigit(*s2))
            *s1++=*s2;
        else
            n++;
        // on avance tjs dans s2
        s2++;
    }
    *s1='\0';
    return n;
}
```

On souhaite représenter une liste d'éléments de type `T` de façon circulaire (le dernier élément est lié au premier) par une structure simplement chaînée.

- 2. Écrivez la déclaration du type `LISTE_CIR` qui permet cette représentation.

```
typedef int T;
struct noeud {
    T elt;
    struct noeud *suivant;
};
typedef struct {
    struct noeud *first, *last;
} LISTE_CIR;
```

- 3. En utilisant cette déclaration de `LISTE_CIR`, dessinez une liste à 0 élément, une liste à 1 élément, une liste à 2 éléments, et enfin une liste à 3 éléments.



- 4. Écrivez la procédure `tourner` qui prend en paramètre une liste `li` et un entier `n` et qui fait `n` fois le tour de la liste de façon circulaire.

```
int estVide(LISTE_CIR l) {
    return l.first == NULL && l.last == NULL;
}

/*
 * Antécédent : n >= 0
 */
void tourner(LISTE_CIR l, int n) {
    assert(n >= 0);
    if (!estVide(l))
        for (int i=0; i<n; i++) {
            struct noeud *q = l.first;
            do
                q = q->suivant;
            while (q != l.first);
        }
}
```

- 5. Écrivez la procédure `ajouterEnTete` qui ajoute un élément au début de la liste circulaire. Cette procédure possède l'en-tête suivant (**que vous devez respecter**) :

```
void ajouterEnTete(LISTE_CIR *li, T e)

void ajouterEnTete(LISTE_CIR *li, T e) {
    struct noeud *p = malloc(sizeof(struct noeud));
    p->elt = e;
    if (estVide(*li)) {
        (*li).first = (*li).last = p;
        p->suivant = p;
    }
    else {
```

```

    p->suivant = (*li).first;
    (*li).first = p;
    (*li).last->suivant = (*li).first;
}
}

```

-
- 6. Donnez la déclaration d'une structure `Date` pour représenter une date formée de 3 entiers : jour, mois et année.

```

typedef struct {
    int jour, mois, annee
} Date;

```

-
- 7. Un fichier de texte contient une suite d'entiers séparés par un ou plusieurs espaces. Trois entiers consécutifs représentent une date. Écrivez la fonction `creerFichDates` qui lit un fichier de texte contenant la suite d'entiers, et qui crée un fichier de dates valides (de type `Date`). Sans l'écrire, vous pourrez utiliser la fonction booléenne `dateValide` qui teste si la date (de type `Date`) passée en paramètre correspond à une date valide ou pas. La fonction `creerFichDates` prend les noms des deux fichiers en paramètre. Son en-tête est le suivant :

```

void creerFichDates(char *in, char *out);

```

```

void creerFichDates(char *in, char *out) {
    FILE *fdIn, *fdOut;

    if ((fdIn = fopen(in, "r")) == NULL) {
        perror(in);
        exit(errno);
    }
    if ((fdOut = fopen(out, "w")) == NULL) {
        perror(out);
        exit(errno);
    }
    // ok : les 2 fichiers sont ouverts
    int j, m, a;
    while ((fscanf(fdIn, "%d %d %d", &j, &m, &a))==3) {
        assert(n==3);
        Date d = {j, m, a};
        if (dateValide(d))
            fwrite(&d, sizeof(Date), 1, fdOut);
    }
    //
    fclose(fdIn);  fclose(fdOut);
}

```