

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. À l'aide d'un **typedef**, déclarez la structure **point** pour représenter les coordonnées (x, y) de type **double** du plan cartésien.

Question sur 1 pt

```
typedef struct {  
    double x, y;  
} point;
```

- 2. À l'aide d'un **typedef**, déclarez la structure **rectangle** pour représenter les coordonnées des points **ptHautGauche** et **ptBasDroit** d'un rectangle dans le plan cartésien.

Question sur 1 pt

```
typedef struct {  
    point ptHautGauche, ptBasDroit;  
} rectangle;
```

- 3. Écrivez la fonction **surface** qui renvoie la surface d'un rectangle passé en paramètre. On considère que le rectangle est aligné avec les axes.

Question sur 2 pts

```
/*  
 * rôle : renvoie la surface du rectangle r  
 */  
double surface(const rectangle r) {  
    double largeur = r.ptBasDroit.x - r.ptHautGauche.x;  
    double hauteur = r.ptHautGauche.y - r.ptBasDroit.y;  
    return largeur*hauteur;  
}
```

- 4. En utilisant uniquement la notation pointeur, et SANS UTILISER de fonctions de **string.h**,

écrivez la fonction **inverser** qui inverse les caractères d'une chaîne de caractères **s** passée en paramètre. La fonction renvoie la chaîne **s** inversée. Par exemple, si **s** égale à **"123456"**, après inversion, **s** sera égale à **"654321"**.

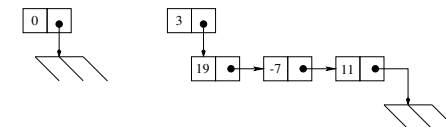
Question sur 5 pts

```
/*  
 * Rôle : échange les valeurs des caractères a et b  
 */  
void échanger(char *a, char *b) {  
    char aux=*a;  
    *a=*b;  
    *b=aux;  
}  
  
/*  
 * Rôle : renvoie la chaîne s dont les caractères ont été inversés  
 * e.g. "123456" => "654321"  
 */  
char *inverser(char *s) {  
    char *p=s;  
    // se placer à la fin de s  
    while (*p) p++;  
    // échanger les caractères de s en parcourant  
    // s par ses deux extrémités  
    char *q=s;  
    while (--p>q)  
        échanger(p, q++);  
    //  
    return s;  
}
```

Une liste linéaire est représentée par une structure dynamique simplement chaînée donnée par les déclarations de types suivantes :

```
typedef ?? T;  
typedef struct noeud {  
    T elt;  
    struct noeud *suivant;  
} *pnoeud;  
typedef struct {  
    int lg;  
    pnoeud tête;  
} liste;
```

avec ces déclarations, on représente, par exemple, la liste vide **<>** et celle d'entiers **< 19, -7, 11 >** comme suit :



- 5. Écrivez la fonction **listeVide** qui renvoie une liste vide.

Question sur 1 pt

```
/*
 * rôle : renvoie une liste vide
 */
liste listeVide(void) {
    return (liste) { 0, NULL };
}
```

- 6. Écrivez la procédure `supprimerEnQueue` qui supprime le dernier élément d'une liste `l`. Cette procédure n'a qu'un seul paramètre la liste `l`.

Question sur 5 pts

```
/*
 * antécédent : l liste non vide
 * rôle : supprime le dernier élément de la liste l
 */
void supprimerEnQueue(liste *l) {
    assert(l->lg!=0);
    // la liste contient au moins un élément
    pnoeud q=l->tête;
    if (l->lg==1)
        // cas particulier : la liste ne contient qu'un élément
        l->tête=NULL;
    else {
        // la liste contient au moins deux éléments
        pnoeud p;
        do {
            p = q;
            q=q->suivant;
        } while (q->suivant!=NULL);
        // q pointe sur l'élément à supprimer
        // et p sur son prédécesseur
        p->suivant = NULL;
    }
    //
    l->lg--;
    // libérer la mémoire occupée par *q
    free(q);
}
```

Un fichier (**attention : qui n'est pas un fichier de texte!**) contient successivement un entier (de type `int`) et une suite de caractères de longueur quelconque (éventuellement vide). L'entier correspond au nombre de caractères de la suite. Par exemple, le fichier suivant, contient 4 entiers et 4 suites de caractères (dont une est vide).

4	b	i	b	i	10	X	i	d	i	a	n	X	i	a	n	0	5	i	n	f	o	C
---	---	---	---	---	----	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

- 7. Écrivez la fonction `longueur` qui lit un fichier organisé comme indiqué ci-dessus, et renvoie le nombre de caractères qu'il contient (19 dans l'exemple précédent). Notez que le fichier peut être vide, et est sans erreur. Réfléchissez bien où mettre chaque suite de caractères lue. L'en-tête de cette fonction est le suivant :

```
/*
 * antécédent : ...
 * rôle : ...
 */
int longueur(const char *nomFich)
```

Question sur 5 pts

```
/*
 * antécédent : nomFich nom du fichier à lire
 * rôle : renvoie la somme des longueurs des suites de caractères
 *         contenues dans le, fichier.
 */
int longueur(const char *nomFich) {
    FILE *fp;
    // ouvrir le fichier en lecture
    if ((fp=fopen(nomFich, "r"))==NULL) {
        perror(nomFich);
        exit(errno);
    }
    // le fichier est ouvert en lecture
    int n, lg=0;
    while (fread(&n, sizeof(int), 1, fp)>0)
        if (n!=0) {
            // allouer dynamiquement un tableau de n caractères
            char suiteCar[n];
            // lire les n caractères
            fread(suiteCar, 1, n, fp);
            // incrémenter la longueur
            lg+=n;
        }
    // fermer le fichier
    fclose(fp);
    //
    return lg;
}
```