Perfectionnement à la programmation en C

Examen – Licence 2 Informatique 2023 – 2024 — *vendredi 24 mai 2024*

Deux feuilles A4 de notes manuscrites personnelles autorisées, tout autre document interdit.

Ce devoir est constitué de 15 questions. Voici quelques conseils :

— lire l'intégralité du sujet avant de commencer

— respecter les **conventions** étudiées. Une fonction qui "marche" peut être incorrecte

— lorsqu'il est demandé de définir une fonction, seuls les paramètres principaux sont précisés. Il est ainsi possible d'**ajouter d'autres paramètres**

— les seules fonctions externes, types et macro-définitions qu'il est autorisé d'utiliser sont ceux apportés par les **en-têtes** `stdio.h`, `stdlib.h` et `assert.h`

— il est autorisé de déclarer des **types** et de définir des **fonctions intermédiaires** si besoin est

---

[Q1] Pour chaque ligne du programme de gauche, **lister** les éventuels effets secondaires et **donner** la valeur et le type de l'expression sous-jacente (s'il y en a une). **Indiquer en justifiant** si les fonctions définies à droite sont à effets secondaires.

```c
int a, *p, t[5];
if (plus5(2) < 6) {
    a = 4;
} else {
    a = 2;
}
p = index(t, a);
p[1] = 0;
```

```c
int *index(int *x, int y) {
    return &x[y];
}
int plus5(int a) {
    int b;
    b = 5;
    return a + b;
}
```

[Q2] **Donner** les erreurs présentes dans cet extrait de code, **indiquer** s'il compile et **proposer** une correction. *Justifier chaque réponse.*

```c
// Renvoie le tableau des sommes cumulées des éléments du tableau passé en
    paramètre
// exemple: [1, 4, 6] -> [1, 5 (1+4), 11 (1+4+6)]
int * sommes_cumulees(int tab[]) {
    int resultat[TAILLE_MAX];     // Tableau des résultats
    int taille = sizeof(tab);     // Taille de tab

    assert(taille < TAILLE_MAX);

    resultat[0] = tab[0];
    for (int i = 1; i =! taille; i++) {
        resultat[i] = resultat[i-1] * tab[i];
    }
    return resultat;
}
```

[Q3] Soit un projet constitué de quatre modules `A`, `B`, `C`, `D` et d'un module principal `Main`. Il figure dans ce projet les inclusions suivantes :

— `Main.c` inclut `B.h` et `C.h`
— `C.h` inclut `B.h`
— `B.c` inclut `C.h`
— `A.h` inclut `C.h` et `D.h`
— `B.h` inclut `D.h`
— `D.h` inclut `C.h`
— `A.c` inclut `D.h`

Tracer le graphe d'inclusions (étendues) du projet.

[Q4] **Expliquer** si le projet contient une ou plusieurs incohérences/erreurs

[Q5] **Écrire** un Makefile *simple* pour compiler le projet selon les bonnes pratiques du cours (on suppose ici que le projet ne présente aucune incohérence)

[Q6] Après une compilation (qu'on supposera sans erreur), on modifie le fichier `B.h`, quels sont les fichiers à produire à nouveau à l'aide de `gcc/clang` ? **Justifier**.

[Q7] **Écrire** une fonction `nom_base` paramétrée par une chaine de caractères correspondant au chemin d'accès d'un fichier. La fonction renvoie l'adresse de la sous-chaine de caractères correspondant au nom du fichier. Par exemple, avec le paramètre `"/home/padawan/perfc/tp1.c"`, la fonction renvoie l'adresse de la sous-chaine `"tp1.c"`. Si le chemin est vide ou termine par un `'/'`, la fonction renvoie l'adresse de la sous-chaine vide `""`.

[Q8] **Écrire** une fonction `compte_suite_aB` à gestion d'erreur paramétrée par un nom de fichier `nom` et qui renvoie le nombre d'occurences de la suite de lettres `aB` dans le fichier. **Documenter** la fonction.

[Q9] **Écrire** une fonction `miroir_8bit` paramétrée par un nombre de 8 bits et renvoie le miroir bit à bit sur 8 bits. Par exemple, la fonction appelée avec le nombre `01011000` renvoie le nombre `00011010`.

[Q10] Sans utiliser de boucle, **écrire** une fonction `bits_consecutifs` paramétrée par un nombre de 64 bits et renvoie un nombre de 64 bits dont chaque bit en position `i` vaut `1` si les bits en position `i` et `i+1` sont de même valeur et `0` sinon. Par exemple, la fonction appelée avec le nombre `11...1101100` renvoie le nombre `01...100101`.
Note : le choix de la valeur du 64e bit du résultat (le bit de poids fort) est laissé libre.

[Q11] **Déclarer** un pointeur `f_1` sur une fonction paramétrée par deux caractères et qui renvoie un entier. **Déclarer** un pointeur `f_2` sur une fonction paramétrée par deux entiers et qui renvoie un pointeur sur une fonction de même prototype que celui de `f_1`. **Donner** une expression ou une instruction correcte utilisant `f_1` et `f_2`.

[Q12] **Écrire** une fonction `separe` paramétrée par un tableau d'entiers de type `int` et un pointeur sur une fonction `test` à type de retour `int` et à un paramètre de type `int`. La fonction renvoie un tableau dont les premiers éléments sont les éléments pour lesquels `test` renvoie `0`, et les derniers sont les autres éléments.

[Q13] **Définir** un type `NoeudGenerique` permettant de représenter un noeud d'un arbre binaire générique, c'est-à-dire dont le type des valeurs n'est pas fixé. **Donner en justifiant** le prototype d'une fonction permettant d'afficher les valeurs d'un arbre binaire générique.

[Q14] En utilisant la fonction `miroir_8bit` (supposée correctement implémentée), **proposer** une fonction permettant de modifier une zone mémoire spécifiée par son adresse et sa taille en son miroir bit à bit.

[Q15] **Écrire** une fonction `main` permettant de tester les fonctions `nom_base` et `miroir_8bit`.