ArcSys1: Systèmes et Réseaux

Examen de C 2023 (1h 30)

Tous documents interdits à l'exception d'un A4 recto manuscrit de votre main et du pense-bête du C fourni en cours. Calculatrices, téléphones, ordinateurs et montres connectées interdites. La correction tiendra compte de la qualité de l'argumentaire et de la présentation. Un code illisible et/ou incompréhensible est un code faux.

★ Exercice 1: Chaînes de caractères et fichiers (6 pts).

▷ **Question 1** (2pt)**:** Écrivez une fonction `char plus_frequent(char* str)`, qui prend en paramètre la chaîne de caractères à analyser, et retourne la lettre la plus fréquente. Par exemple, `plus_frequent("␣Bon␣courage␣!␣␣")` doit retourner `'o'`. Vous pouvez supposer qu'il n'y a pas de lettre accentuée dans le texte.

▷ **Question 2** (2pt)**:** Écrivez un programme complet plus-frequent prenant une chaîne de caractères à analyser en premier argument de la ligne de commande, et affichant le résultat de la manière suivante :

```
$ ./plus-frequent " Bon courage !"
La lettre la plus fréquente est 'o'.
```

▷ **Question 3** (2pt)**:** Écrivez un programme somme prenant un nombre arbitraires d'entiers sous la forme d'arguments de la ligne de commandes, et en affiche la somme. Vous pouvez supposer que les paramètres ne sont que des nombres comme attendu. Par exemple :

```
$ ./somme 5 2 11 24
La somme est : 42
```

★ Exercice 2: Mémoire dynamique : liste simplement chaînée (8pts).

On souhaite représenter une liste simplement chaînée d'entiers. Chaque cellule de la liste est représenté par une structure C comprenant deux champs: un entier représentant la valeur de cette cellule, et un champ `next` représentant la cellule suivante dans la liste, ou `NULL` pour la dernière cellule de la liste.

▷ **Question 1** (2pt)**:** Écrivez la structure nécessaire, ainsi que lafonction `N` permettant de créer une nouvelle cellule en mémoire dont la valeur et la suite sont passées en paramètre. Donnez ensuite le code permettant de créer la liste dont les valeurs successives sont 27, 14, 34, 66 et 58 (dans cet ordre).

```
struct list * N(int val, struct list * next);
```

▷ **Question 2** (2pt)**:** Écrivez les fonctions `list_len` (retournant la longueur d'une liste), `list_max` (retournant la valeur maximale d'une liste donnée, ou `INT_MIN` si la liste est vide), `list_concat` (modifiant la première liste passée en paramètre pour ajouter la seconde à la fin, et retournant la liste résultante de la concaténation) et `list_free` (libérant la mémoire prise par une liste).

```
int list_len(struct list * l);
int list_max(struct list * l);
struct list * list_concat(struct list * l1, struct list * l2);
void list_free(struct list * l);
```

▷ **Question 3** (2pt)**:** Écrivez la fonction `list_insert` permettant d'ajouter un élément dans une liste triée. On supposera que la liste est triée avant l'insertion, et on veillera à ce que le nouvel élément soit bien placé dans la liste. La fonction retournera le début de la liste modifiée, contenant toutes les cellules pré-existantes plus la nouvelle créée pour l'élément inseré.

```
struct list * list_insert(struct list * l, int val);
```

▷ **Question 4** (2pt)**:** Écrivez la fonction `list_sort` prenant une liste en paramètre et retournant une nouvelle liste triée contenant toutes les valeurs de la liste d'entrée. Vous implémenterez un tri par insertion en parcourant toutes les valeurs de la liste d'entrée et en les insérant les unes après les autres dans la liste résultat en construction.

★ **Exercice 3: Mémoire statique en C (6 pts).**

Tous les programmes de cet exercice compilent et ne provoquent pas d'erreur à l'exécution (même si on utilise `valgrind` ou `gdb`).

fichier1.c

```
1  #include <stdio.h>
2
3  static int i = 6;
4
5  int f() {
6    static int i = 1;
7    return ++i;
8  }
9
10 int g() { return i++; }
11
12 int main(void) {
13   printf("g() 1ere fois %d\n", g());
14   printf("f() 1ere fois %d\n", f());
15   printf("g() 2ieme fois %d\n", g());
16   printf("f() 2ieme fois %d\n", f());
17 }
```

fichier2.c

```
1  #include <stdio.h>
2  int main() {
3    int a[] = {1, 2, 4, 8, 16};
4    int *p[2] = {&a[2], &a[1]};
5    p[0] += *(p[1]);
6    printf("%d\n", **p);
7  }
```

fichier3.c

```
1  #include <stdio.h>
2  int main() {
3    char s[] = "chaine";
4    char *p = s + 1;
5    p++;
6    *p = *s + 1;
7    printf("%s", p);
8  }
```

▷ **Question 1** (2pt)**:** Indiquez la sortie affichée lorsqu'on exécute le programme `fichier1.c`, en justifiant par un schéma mémoire global du processus.

▷ **Question 2** (2pt)**:** Discutez de la même façon la sortie du programme `fichier2.c` avec un schéma des variables locales avant et après la ligne 5.

▷ **Question 3** (2pt)**:** Discutez de la même façon la sortie du programme `fichier3.c` avec les schémas mémoire nécessaires, et expliquez pourquoi ce programme produit une erreur mémoire lorsqu'on remplace la ligne 3 par la suivante: `char* s = "chaine"` (le type de `s` change).