Examen final de programmation impérative

ÉNSIIE, semestre 1

mercredi 19 janvier 2022

Durée : 1h45.
Tout document papier autorisé. Aucun appareil électronique autorisé (sauf dérogation par la scolarité).
Ce sujet comporte 3 exercices indépendants, qui peuvent être traités dans l'ordre voulu. Il contient 5 pages. Le barème est donné à titre indicatif, il est susceptible d'être modifié. Le total est sur 20 points. Il va de soi que toute réponse devra être justifiée, et que toute fonction devra être commentée (au minimum `require`, `assigns` et `ensures`).

Exercice 1 : Fonctions simples (6 points)

1. Proposer une procédure dont l'effet est d'incrémenter la valeur d'une variable de type `int` donnée en paramètre par une valeur également passée en paramètre. (Exemple de cas d'utilisation : la fonction est définie *avant* `main`. À l'intérieur de `main`, une variable est définie et la fonction appelée avec cette variable et une constante. On n'écrira pas la fonction `main`.)
2. Proposer une fonction qui prend en paramètre un tableau de `double` et sa taille et qui retourne la moyenne des valeurs contenues dans le tableau.
3. Soit le type suivant :

   ```
   struct couple {
     int left;
     int right;
   };
   ```

   Proposer une fonction qui prend en paramètre un entier n et retourne un tableaux de `struct couple` de taille n dans lequel à la position i on aura dans `left` la somme des entiers de 1 à i et dans `right` le produit des entiers de 1 à n.

   NB : à la position 0, on aura donc respectivement 0 et 1, les éléments neutres pour l'addition et la multiplication.
4. Proposer une procédure qui prend en paramètre une chaîne de caractère et qui l'encode (en place) à l'aide du chiffre de César : chaque lettre est remplacée par celle située 13 positions après dans l'ordre alphabétique (en revenant à a après z). On supposera que la chaîne ne contient que des lettres minuscules. Par exemple la chaîne `"cesar"` sera transformée en `"prfne"`.

   Indication : le code ASCII de `'a'` est 97.
5. Proposer une fonction qui prend en paramètre deux chaînes de caractères et qui teste si la première est une sous-chaîne de l'autre. Par exemple, avec les chaînes

`"World"` et `"Hello,␣World!"` on retournera 1, tandis qu'avec `"Hlo␣ol!"` et `"Hello,␣World!"` on retournera 0.

6. Proposer une fonction qui prend en paramètre une chaîne de caractères et deux entiers i et j, et qui retourne une copie de la sous-chaîne contenue entre les positions i inclue et j exclue. Par exemple, avec la chaîne `"Hello,␣World!"`, `i` $= 7$ et `j` $= 12$ on retournera une chaîne contenant `"World"`.

Exercice 2 : Listes (7 points)

On considère les listes telles que vues en cours, qui contiennent des entiers. En particulier on utilisera les fonctions et procédures `cons(n, l)` qui retourne un nouveau maillon contenant `n` et qui pointe vers `l`, `add(pl, n)` qui rajoute un maillon contenant `n` en tête de la liste passée par référence `pl`, et `remove_once(pl, n)` qui retire le premier maillon contenant `n` (s'il en existe un) de la liste passée par référence `pl`. On rappelle le code de cette procédure :

```
void remove_once(list *pl, int n) {
  list tmp, visit;
  if (*pl == NULL) return;
  if ((*pl)->val == n) {
    tmp = *pl;
    *pl = (*pl)->next;
    free(tmp);
  } else {
    visit = *pl;
    while (visit->next != NULL && visit->next->val != n)
      visit = visit->next;
    if (visit->next == NULL) return;
    tmp = visit->next;
    visit->next = visit->next->next;
    free(tmp);
  }
}
```

1. Écrire une fonction `mem` qui prend en paramètre un entier et une liste, et qui teste si l'entier est présent dans la liste.
2. Écrire une procédure `add_last` qui prend en paramètre une liste par référence et un entier, et qui ajoute un nouveau maillon contenant l'entier à la fin de la liste.

On considère les instructions suivantes :

```
1 list f,g;
2 f = NULL;
3 add(&f, 3);
```

```
4 g = cons(2, f);
5 add(&f, 4);
6 add_last(&g, 1);
7 remove_once(&f,3);
8 printf("%i\n", mem(2,g));
```

3. Représenter l'état de la mémoire à la fin des lignes 2, 3, 4, 5, 6 et 7.
4. Que se passe-t-il à la ligne 8 ?

Exercice 3 : Jeu des 7 erreurs (7 points)

Les procédures et fonctions suivantes ne sont pas correctes par rapport à la spécification donnée. Les corriger en expliquant pourquoi il y a une erreur.

1.
```
/*@ requires m >= 0
    assigns nothing
    ensures returns the kinetic energy given
    mass m and velocity v */
double kinetic_energy(double m, double v) {
  return 1 / 2 * m * v * v;
}
```

2.
```
#include <stdlib.h>

typedef struct node *tree;
struct node {
  int content;
  tree left_child;
  tree right_child;
};

/*@ requires l and r valid trees
    assigns nothing
    returns a new node whose content is c,
    whose left child is l, and whose right
    child is r */
tree node(int c, tree l, tree r) {
  tree res = malloc(sizeof (tree));
  res->content = c;
  res->left_child = l;
  res->right_child = r;
  return res;
}
```

3.
```
/*@ requires t has size s >= 0
    assigns nothing
    ensures returns the mean value of the
    elements of t        */
char mean(char t[], int s) {
  int i;
  char r = 0;
  for (i = 0; i < s; i += 1)
    r += t[i];
  return r / s;
}
```

4. On considère la définition des listes chaînées vue en cours.

```
/*@ requires l1 and l2 are well-formed lists
    assigns potentially one of the buckets of l1
    ensures returns the concatenation of l1 and l2 */
list concat(list l1, list l2) {
  list visit = l1;
  while (visit->next != NULL)
    visit = visit->next;
  visit->next = l2;
  return l1;
}
```

5.
```
struct rat {
  int num;
  int den;
};

/*@ requires r.den != 0
    assigns r
    ensures inverse r by effect */
void inverse(struct rat r) {
  int tmp = r.num;
  r.num = r.den;
  r.den = tmp;
}
```

6.
```
#include <stdio.h>

/*@ requires t is a valid array
    assigns nothing
    ensures print the content of t
    on the standard output   */
```

```
void print_tab(int t[]) {
  int i;
  for (i = 0; i < sizeof(t); i += 1)
    printf("%i␣", t[i]);
}
```

7. Étant donné un entier $n > 0$, la suite de Collatz partant de n est la suite d'entiers dont le premier terme est n et dont les termes successifs sont donnés par $c_{k+1} = \begin{cases} \frac{c_k}{2} & \text{si } c_k \text{ est pair} \\ 3c_k + 1 & \text{si } c_k \text{ est impair} \end{cases}$. La conjecture de Collatz suppose qu'une telle suite atteint forcément la valeur 1 au bout d'un certain temps.

```
/*@ requires c valid address, m > 0
    assigns *c
    ensures add to *c the number of steps needed by the
    Collatz sequence to reach 1 starting from m */
void Collatz(int m, int *c) {
  if (m > 1) {
    *c += 1;
    if (m % 2 = 0)
      Collatz(m / 2, c);
    else
      Collatz(3 * m + 1, c);
  }
}
```