Paradigmes et interprétation - partie 1/3

Examen session 1 2022-2023

L2/L3 informatique – Université Côte d'Azur

Durée: 40 minutes (1/3 de 2 heures).

- Aucun document ni aucune machine ne sont autorisés. Un mémo est donné en fin de sujet.
- Les téléphones doivent être rangés.
- Vous pouvez déborder du cadre de réponses si besoin, sans écrire sur les cases réservées à la correction en haut du cadre.
- Les réponses doivent être écrites lisiblement, en respectant autant que possible l'interligne proposé. Le correcteur blanc et l'effaceur sont autorisés.
- Les questions sont indépendantes: une question peut être sautée, et une fonction `f` demandée à une question peut être utilisée pour définir une fonction `g` dans une question ultérieure même si la solution pour `f` n'a pas été trouvée.
- Le barème est une estimation basse, il pourra être revu à la hausse (note > 20 possible dans ce cas). Le nombre de points d'une question est proportionnel à l'estimation de la difficulté de la question.

☐0 ☐0 ☐0 ☐0 ☐0 ☐0 ☐0 ☐0
☐1 ☐1 ☐1 ☐1 ☐1 ☐1 ☐1 ☐1
☐2 ☐2 ☐2 ☐2 ☐2 ☐2 ☐2 ☐2
☐3 ☐3 ☐3 ☐3 ☐3 ☐3 ☐3 ☐3
☐4 ☐4 ☐4 ☐4 ☐4 ☐4 ☐4 ☐4
☐5 ☐5 ☐5 ☐5 ☐5 ☐5 ☐5 ☐5
☐6 ☐6 ☐6 ☐6 ☐6 ☐6 ☐6 ☐6
☐7 ☐7 ☐7 ☐7 ☐7 ☐7 ☐7 ☐7
☐8 ☐8 ☐8 ☐8 ☐8 ☐8 ☐8 ☐8
☐9 ☐9 ☐9 ☐9 ☐9 ☐9 ☐9 ☐9

← Codez ci-contre votre numéro d'étudiant : cochez dans la première colonne le premier chiffre (a priori un 2), puis dans la deuxième colonne le deuxième chiffre, etc

Date de naissance ou pseudo[a]:

. . . . . .

[a] utilisé en cas de problème pour vous retrouver avec le numéro d'étudiant. Merci de garder l'anonymat et de ne pas donner un pseudo qui permet de vous identifier

Erreurs mémoires (2 points)

On considère le programme suivant

```
let x = Box::new(Box::new(0)); free(*x); *x
```

Question 1 (0,5 pts) Quel(s) type(s) d'erreur(s) contient ce programme?

- ☐ pointeur pendant (dangling pointer)
- ☐ violation mémoire (lecture à une adresse non allouée)
- ☐ fuite mémoire (memory leak)

Question 2 (0,5 pts) Même question pour le programme suivant.

```
let x = Box::new(Box::new(0)); free(x); *x
```

- ☐ fuite mémoire (memory leak)
- ☐ violation mémoire (lecture à une adresse non allouée)
- ☐ pointeur pendant (dangling pointer)

Question 3 (1 pts) Même question pour le programme suivant.

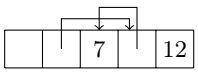
```
let x = Box::new(0); {let x = Box::new(1)}; free(x); *x
```

- ☐ violation mémoire (lecture à une adresse non allouée)
- ☐ pointeur pendant (dangling pointer)
- ☐ fuite mémoire (memory leak)

Gestion mémoire (2 points)

On souhaite écrire en Python un interpréteur pour un petit langage jouet avec des boites comme vu au cours 3. Les boites peuvent contenir des entiers ou des pointeurs vers d'autres boites. On décide de représenter le tas par une liste Python de sorte que la cellule à l'adresse i correspond au ième élément de la liste. Si la cellule n'est pas allouée, on la représente par None, si elle est un pointeur vers une adresse `a`, on la représente par une liste de taille 1 contenant `a`, et si c'est un entier on y met directement cet entier. Par exemple la liste Python suivante correspond au tas représenté ci-contre.

```
heap = [None, [3], 7, [2], 12]
```



Question 4 (1 pts) Ecrire une fonction `alloc(heap, v)` qui alloue une cellule non allouée et renvoie son adresse. Par exemple si `heap` est comme ci-dessus, `alloc(heap, 42)` le modifie en `heap = [42, [3], 7, [2], 12]`.

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

Pour votre examen, imprimez de préférence les documents compilés à l'aide de auto-multiple-choice.

Question 5 (1 pts) Ecrire une fonction `drop(heap, a)` qui désalloue l'adresse a et toutes celles auxquelles elle donne accès en suivant les pointeurs, récursivement. Par exemple, si `heap = [42, [3], 7, [2], 12]`, `drop(heap,1)` le modifiera en `heap = [42, None, None, None, 12]`

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

.....

.....

.....

.....

Qui rase Figaro? (2 points)

On considère le programme Datalog suivant.

```
1.  rase(almaviva, basile).  % almaviva rase basile
2.  rase(cherubin, cherubin).  % cherubin se rase tout seul
% figaro rase ceux qui ne se rasent pas (qu')eux-même
3.  rase(figaro, X) :- rase(X, Z), diff(X, Z).
4.  diff(X, X) :- !, fail.
5.  diff(X, Y).
% requête:  qui rase Figaro?
?- rase(X, figaro)
```

Lorsqu'on l'exécute, on observe qu'un premier résultat est renvoyé, puis le programme part dans une recherche infinie.

Question 6 (2 pts) Dessinez l' arbre de recherche de la requête jusqu'à la feuille de succès (ne cherchez pas à développer le début de la branche infinie), et précisez la réponse à la requête, autrement dit répondez à la question: qui rase Figaro? Précisez à chaque étape la ligne de programme et les substitutions utilisées, ainsi que les branches coupées (voir exemple dans le mémo). Pour économiser l'espace, vous pouvez abréger les identifiants en indiquant seulement la première lettre (ex: `r(a,f)` au lieu de `rase(almaviva, figaro)`).

☐0 ☐1 ☐2 ☐3 ☐4 ☐5 *Cases réservées à la correction*

Memo sur Python

```
>>> L = [1,2,3]
>>> L[0]
1
>>> L[-1]
3
>>> for i in range(1, len(L)+1):
        print(L[-i])
3
2
1
>>> D = {'a':1, 'b':2, 'c':3}
>>> D['a']
1
>>> 'a' in D
True
>>> 'z' in D
False
```

Memo sur le langage du cours 2 avec désallocation explicite

```
imp # let x = Box::new(1)
x : Box<int> = @0(1)
imp # #dump_heap
[1, ∅, ∅, ∅]
imp # free(x)
- : unit = ()
imp # #dump_heap
[∅, ∅, ∅, ∅]
imp # *x = 2
Evaluation Error : dangling pointer error: address 0 is not allocated
imp # free(x)
Evaluation Error : double free or dangling pointer error: trying to free 0 that is not allocated
```

Memo sur Prolog On numérote les lignes par comodité.

```
1. a(1,1).
2. a(2,2).
3. b(X,Y) :- a(X,X), a(Y,Y), a(X,Y), !.
4. ?- b(Z,T).
```