

IENAC S02
Programmation
Durée : 15mn
Sans document

Ce sujet comporte 4 pages.

Barème pour chaque question (**une seule réponse par question**¹) :

- bonne réponse : 1 point ;
- mauvaise réponse : $-\frac{1}{3}$ de point ;
- pas de réponse (ou trop de réponses) : 0.

1. Le style impératif consiste essentiellement à
 - faire des appels de fonctions
 - faire des boucles **while**
 - écrire et lire en mémoire, en séquence
 - utiliser des références
2. Lequel de ses identificateurs n'est pas lexicalement correct
 - toto_encore
 - bob.txt
 - f1
 - _loop
3. Qu'affiche le programme suivant

```
void f(int x) {  
    int x = 1729;  
    { int y = x; y = 12; }  
    printf("%d", y);  
}  
main() {  
    f(123);  
}
```

- 1729
- 12
- 123
- Rien

4. Qu'affiche le programme suivant

```
let x = ref 1;;  
let f = fun () -> let x = ref 2 in x := 3;;  
let g = fun () -> x := 4; let x = ref 5 in ();;  
f (); g (); Printf.printf "%d\n" !x;;
```

- 2
- 3
- 4
- 5

¹Ce qui ne signifie pas une seule **bonne** réponse par question.

5. Le message suivant

```
warning: 'return' with a value, in function returning void
```

est affiché lors de la compilation avec `gcc -Wall` pour la fonction

- `void f() { return 1; }`
- `int g() {}`
- `char h() { return 7.0; }`
- aucune de ces fonctions

6. La fonction suivante

```
let f = fun x y -> if x = 1 then !y else float x;;
```

est de type

- `int -> float ref -> float`
- `(int -> float) -> float ref -> int`
- `int -> float -> float`
- n'est pas typable

7. La fonction suivante

```
char *f(char g(float x), int y) {  
    return &g((float)y);  
}
```

- est syntaxiquement incorrecte
- est mal typée
- n'est pas compilable
- est correcte

8. La fonction suivante

```
let rec f = fun f x -> if x = 0 then f x else x;;
```

- est récursive non terminale
- est récursive terminale
- n'est pas récursive
- n'est pas correcte

9. Soit le programme suivant

```
int t[3] = { 7 , 8 , 9 };
```

Que vaut `t[2]+t[3]`

- 5
- 17
- 9
- ça dépend

10. Qu'affiche le programme suivant ?

```
let m = Array.create 10 (Array.create 5 7);;  
for i = 0 to 4 do m.(2).(i) <- i+1 done;;  
Printf.printf "%d\n" m.(1).(3);;
```

- 7
- 4
- 3
- le programme ne peut pas être compilé

11. Soit la fonction suivante

```
let rec iter = fun f n z ->
  if n = 0 then z else f (iter f (n-1) z);;
```

La fonction « puissance de 2 » s'écrit

- `fun n -> iter (fun x -> 2 * x) 1 n`
- `fun n -> iter (2 * n) n 1`
- `fun n -> iter (fun x -> 2 * x) n 1`
- ne peut pas s'écrire avec `iter`

12. Les éléments d'un tableau de n éléments

- ne peuvent être que des entiers entre 0 et $n - 1$
- sont indexés de 0 à n
- sont éventuellement de type différent
- sont accessibles en temps constant

13. La fonction suivante

```
let f = fun n ->
  let s = malloc(n) in
  for i = 0 to n - 1 do s.(i) = i done;
  s;;
```

- renvoie un tableau d'entiers
- n'est pas syntaxiquement correcte
- n'est pas typable
- ne veut rien dire

14. Soient les fichiers suivant

```
/* toto.c */
static int x = 1;
extern void f(int);
int g(int y) { f(y); return(x+y);}
```

```
/* titi.c */
extern int g(int);
int main() { return g(91); }
```

La commande de compilation `gcc -Wall -c toto.c titi.c`

- s'exécute correctement
- produit le message `undefined symbol f`
- génère un exécutable `a.out`
- ne fonctionne pas car il manque `titi.h` et `toto.h`

15. Pour représenter un point à la surface de la terre de l'hémisphère N, le meilleur type en **O'Caml** est le suivant

- `type t = Lat of float | Long of float`
- `type t = float * float`
- `type t = { lat : float; long : float }`
- `type t = { lat : float >= 0; long : float }`

16. En **C**, un type produit se construit avec

- `union`
- `enum`
- `struct`
- `typedef`

17. Qu'affiche le programme suivant

```
int main() {
    int x = 2;
    switch (x) {
        case 1: x = 4;
        case 2: x = 5;
        case 3: x = 6;
        default: {}
    }
    printf("%d\n", x);
    return 0;
}
```

- 2
- 5
- 6
- rien car il n'est pas correct

18. Soit la fonction suivante

```
let rec f = fun l ->
    match l with
    | [] -> []
    | x::xs -> l @ f xs;;
```

Que renvoie l'appel f [1; 2; 3]

- rien car ça boucle
- [3; 2; 1]
- [[1; 2; 3]; [2; 3]; [3]]
- [1; 2; 3; 2; 3; 3]

19. Soient les déclarations classiques suivantes pour représenter une liste

```
struct cons { int car ; struct cons* cdr; };
typedef struct cons* list;
const list nil = (list)0;
```

La fonction suivante appliquée à la liste des 10 premiers entiers

```
int d(list l) {
    while(l)
        l = l->cdr;
    return l->car;
}
```

- n'est pas compilable
- retourne le dernier élément d'une liste
- produit un Segmentation fault
- boucle indéfiniment

20. Que vaut l'expression suivante

```
List.fold_right (fun x r -> x::x*x::r) [1;2;3] [4]
```

- [1;1;2;4;3;9;4]
- [(1,1);(2,4);(3,9);(4,16)]
- [1;1;2;4;3;9]
- [4]

21. Question subsidiaire. Soit la fonction

```
let a b=let rec a b c d=if c=0 then d else
b(a b(c-1)d)in a(fun c b->a c(b+1)1)b succ;;
```

Que retourne (a 3 3) ?