

Exercice 1

1/ quand on évalue 33 (“33;;” dans l’interpréteur), l’interpréteur nous affiche “- : int = 33”. On ne peut pas utiliser les touches directionnelles.

2/ Les informations données sont exactement les mêmes, mis à part le fait qu’on puisse utiliser les touches directionnelles pour donner l’évaluation précédente.

3

- a) “2;;”
 - a) moi : int
 - b) toplevel : int
 - c) no comment
- b) “2.0;;”
 - a) moi : float
 - b) toplevel : float
 - c) no comment
- c) “2,0;;”
 - a) moi : float
 - b) toplevel : int * int
 - c) la virgule signifie un couple de valeur
- d) “2;0;;”
 - a) moi : int, int
 - b) toplevel : int et “this expression should have type unit”
 - c) ; est utiliser généralement pour finir les fonctions, il s’attend à ce que le ‘2’ ait le bon type.
- e) “(2,0);;”
 - a) moi : int * int
 - b) toplevel : int * int
 - c) équivalent à 2,0
- f) “(2;0);;”
 - a) moi : int et “this expression should have type unit”
 - b) toplevel : int et “this expression should have type unit”
 - c) équivalent à 2;0
- g) “a;;”

- a) moi : “this element has not been declared”
 - b) toplevel : “unbound value”
 - c) cet alias n’a pas été créer
- h) “’a’;;”
- a) moi : char
 - b) toplevel : char
 - c) no comment
- i) “”a”;;”
- a) moi : string
 - b) toplevel : string
 - c) no comment
- j) “true;;”
- a) moi : bool
 - b) toplevel : bool
 - c) no comment
- k) “();;”
- a) moi : void
 - b) toplevel : unit
 - c) l’élément est présent dans le langage mais ne possède pas de type (void)
- l) “[];;”
- a) moi : array
 - b) toplevel : ‘a list
 - c) ces crochets correspondent à une liste vide
- m) “[1];;”
- a) moi : une liste possédant la valeur 1
 - b) toplevel : int list = [1]
 - c) les crochets + des éléments dans les crochets → liste non vide
- n) “[1, true];;”
- a) moi : erreur
 - b) toplevel : (int * bool) list
 - c) Créer une liste de couple d’int et de bool (à cause la virgule)
- o) “[1; true];;”

- a) moi : bool et “this expression should have type unit”
- b) toplevel : This expression has type bool but an expression was expected of type int
- c) le ; est un délimiteur des éléments dans la liste, or une liste ne peut contenir que un seul type d’élément

4/

- a) 1, 1.0
- b) impossible
- c) [“”]
- d) [true], “”
- e) impossible (‘a list * int seulement)
- f) [[1]]

5/

- a) 1+2
 - a) moi : 3
 - b) toplevel : 3
 - c) no comment
- b) 1.1 + 2.2
 - a) moi : 3.3
 - b) toplevel : This expression has type float but an expression was expected of type int
 - c) la somme de 2 flottants se fait grâce à +.
- c) 1.1 + 2
 - a) moi : This expression has type float but an expression was expected of type int
 - b) toplevel : This expression has type float but an expression was expected of type int
 - c) on ne peut pas faire la somme d’un flottant avec un entier
- d) 2 / 3
 - a) moi : 0
 - b) toplevel : 0
 - c) no comment
- e) 7 mod 2
 - a) moi : 1
 - b) toplevel : 1
 - c) no comment

- f) `7. mod 3.`
 - a) moi : 1.
 - b) toplevel : This expression has type float but an expression was expected of type int
 - c) On ne peut pas faire le modulo de 2 nombres flottants
- g) `int_of_float(2. ** 3.)`
 - a) moi : transforme le flottant 2^3 en int
 - b) toplevel : 8
 - c) no comment
- h) `2 = 3`
 - a) moi : false
 - b) toplevel : false
 - c) no comment
- i) `'a' = 'b'`
 - a) moi : false
 - b) toplevel : false
 - c) no comment
- j) `"a" = 'a'`
 - a) moi : erreur de type
 - b) toplevel : This expression has type char but an expression was expected of type string
 - c) on compare 2 types différents
- k) `not 1 = 0`
 - a) moi : true
 - b) toplevel : This expression has type int but an expression was expected of type bool
 - c) l'interpréteur execute d'abord not 1, not s'applique seulement sur le type bool
- l) `not (1=0)`
 - a) moi : true
 - b) toplevel : true
 - c) no comment

6/

`(print_string("true"); true;)&&(print_string("false"); false;);; → truefalse`

`(print_string("false"); false;)&&(print_string("true"); true;);; → false`

`(print_string("true"); true;)||((print_string("false"); false;);; → true`

`(print_string("false"); false;)|(print_string("true"); true;);; → false true`

Exercice 2

1/ `let pi = acos(-1.) and r = 2. in pi *. r**2.;;`

2/ `let pi = acos(-1.) and r = 2. in let p = 2. *. pi *. r and d = r *. 2. and h = 10. in let a = 2. *. d +. p *. h and v = d *. h in (p, a, v);;`

Correction : `let pi = acos (-1.) in let h = 5. and r = 2. in let d = pi *. r *. r and p = 2. *. pi *.r in p, 2. *. d +. p *. h, d *. h;;`

Exercice 3

1/

- a) non, car a est liaison locale
- b) oui, car recréer la liaison b (son type peut donc changer)
- c) oui, car c'est équivalent à écrire `c + c` (possible)
- d) oui, car c'est équivalent à écrire `let e = 1 and f = e` mais aussi parce que la liaison e se fait avant celui de f
- e) idem
- f) non, car le `in` n'est pas une expression
- g) oui

2/ 1, 1.2, 1 et 3

Exercice 4

1/ `a = 1` et `b = 2`

`if a < b then b else a;;`

2/ `if a < b then if c < a then c else a else b;;`

3/ `a` et "odd" ne sont pas de même type donc il y aura une erreur.

4/ on attend un `in` après l'utilisation de `let`, donc erreur au niveau du `else`

`if a < 10 then let b = "small" in b else let b = "large" in b;;`

5/ `if a mod 2 = 1 then a / 2 + 1 else a / 2;;`

6/ `let expr = if a < b then a * a else b * b in if c mod 3 == 0 then expr + 1 else expr;;`

Exercice 5

1/ Cette fonction ne peut pas s'appliquer sur les flottants.

5/ 4, 7, 9, 5