OCaml Quizzes

Quiz 1

Question 1 – The Church Rosser property assumes

A. Mutable variables
B. Immutable variables

Question 2 – For imperative languages (C/C++/Java/…), evaluation order affects program, output

A. False
B. True

Question 3 – For functional languages (OCaml/Haskell/…), evaluation order affects program outputs

A. False
B. True

Question 4 – Select all true statements. Functional languages

A. Have global state
B. Easier to verify than imperative languages
C. Easier to parallelise than imperative languages

Question 5 – Select all companies using OCaml

A. Facebook
B. Bloomberg
C. Jane Street
D. Docker

Question 6 – Infer the type for the expression: *(1 + 1)*

A. float
B. int
C. int -> int -> int
D. bool

Question 7 – What is the type of the OCaml operator: *(+)*

A. float
B. float -> float -> float
C. int -> int -> int
D. int

Question 8 – What is the type for the OCaml operator: *(/.)*

A. int -> int -> int
B. float -> float -> float
C. float
D. int

Question 9 – What is the type of function: *float_of_int*

A. float -> int -> int
B. float -> int -> float
C. int -> float
D. float -> int

Question 10 – What is the type of function: *int_of_float*

A. int -> float
B. float -> int

Question 11 – Infer the type for *'a'*

A. string
B. int
C. char
D. char list

Question 12 – Evaluate: *not (not false)*

A. true
B. false

Question 13 – Apply one beta reduction to: *(λx . ((λ y . x + y) 3)) 6*

A. 9
B. 6 + 3
C. 3 + 6
D. (λy . 6 + y) 3

Quiz 2

Question 1 – Infer the type for expression: *(1 + (3 * 5))*

A. int -> int
B. int
C. int -> int -> int

Question 2 – Infer the type for expression: *(λx . x / 2)*

A. int
B. int -> int
C. float -> float
D. float

Question 3 – Infer the type for expression: *(λx . (λy . (x +. y)))*

A. int
B. int -> int -> int
C. float -> float
D. float -> float -> float

Question 4 – Infer the type for expression: *(λx . (λy . (float_of_int (x + y))))*

A. int -> int -> int
B. float -> float -> float
C. float -> float -> int
D. int -> int -> float

Question 5 – What OCaml function has the type: *float -> float -> float*

A. float_of_int
B. /
C. +.
D. Int_of_float

Question 6 – The following code has an error. Is the error syntactic or semantic? *(func x -> x + y)*

A. Syntactic
B. Semantic

Question 7 – The following code has an error. Is the error syntactic or semantic? *6.0 /. 3*

A. Syntactic
B. Semantic

Question 8 – evaluate: *ceil 4.3*

A. 4.0
B. 5.0
C. 4.3

Question 9 – evaluate: *floor 8.7*

A. 7.0
B. 8.0
C. 8.7

Question 10 – evaluate: *((fun x -> x) 9)*

A. x
B. 9
C. int
D. int -> int

Question 11 – Evaluate: *(fun x -> x + 1) ((fun y -> y *3) 4)*

A. 3
B. 12
C. 15
D. 13

Question 12 – Evaluate: *(fun x -> (x, x)) (ceil 3.1)*

A. (3, 1)
B. 4.0
C. (4.0, 4.0)
D. (3.0, 3.0)

Question 13 – What is the type for OCaml function *(^)*

A. int

B. float -> float

C. string -> string -> string

D. float -> float -> int

Question 14 – Evaluate *not (not (not false)))*

A. False

B. True

Question 15 – Evaluate *true || false*

A. False

B. True

Question 16 – Evaluate *(not false) && false*

A. False

B. True

Question 17 – Evaluate: *(1 = 2) || (3 = 3)*

A. False

B. True

Question 18 – Evaluate: *(fun x -> (fun y -> x && (x || y))) true false*

A. False

B. True

Question 19 – Evaluate: *(4 < 6) && (7 = 7)*

A. False

B. True

Question 20 – Evaluate: *(4.4 > 4.4) || ("OCaml" > "java")*

A. False

B. True

Quiz 3

Question 1 – What is the type of this function: *fun x -> x * 2*

A. int

B. int -> int

C. int -> int -> int

D. float -> float

Question 2 – What will this print when evaluated?

```
let my_function =
  let x = 2 in
   let x = 9 in
    x ;;
```

A. 2

B. 9

Question 3 – What is the domain of this function?

A. string -> int
B. string
C. int

```
let size_of_string s = String.length s ;;
```

Question 4 – Anonymous functions are defined with …

A. let
B. fun

Question 5 – What is the type of: *fun x -> (fun y -> x + y)*

A. int
B. int -> int
C. (int, int) -> int
D. Int -> int -> int

Question 6 – What will *is_zero 1* evaluate to?

A. True
B. False

```
let is_zero (i: int) : bool =
  match i with
    0 -> true
  | n -> false ;;
```

Question 7 – What will *is_zero 0* evaluate to?

A. True
B. False

```
let is_zero (i: int) : bool =
  match i with
    n -> false
  | 0 -> true ;;
```

Quiz 4

Question 1 – What will this return: *categorise_language "prolog"*

A. "functional"
B. "object oriented"
C. "logic"

```
let categorise_language : string -> string =
  fun language ->
    match language with
      "ocaml" -> "functional"
    | "Haskell" -> "functional"
    | "java" -> "object-oriented"
    | "prolog" -> "logic" ;;
```

Question 2 – What will this return: *categorise_person 20*

A. "teenager"
B. "child"
C. "adult"

```
let categorise_person : int -> string =
  fun age ->
    match age with
      n -> "teenager"
    | n when n < 13 -> "child"
    | n when n > 19-> "adult"
```

Question 3 – How many recursive calls will this function make to itself for: *factorial 3*

A. 1
B. 2
C. 3

```
let rec factorial n =
  match n with
   0 -> 1
 | i -> i * factorial (i-1) ;;
```

Question 4 – What is the head of: *[1; 2; 3]*

A. [2; 3]
B. 1
C. [1]

Question 5 – What is the rest in: *let (x :: rest) = [1; 2; 3]*

A. 3
B. [3]
C. 1
D. [2; 3]

Quiz 5

Question 1 – What is returned by: *head [6; 5; 2]*

A. [6]
B. 6
C. [5; 2]
D. [2]

Question 2 – What is returned by: *tail [3; 1; 2]*

A. 3
B. [1; 2]
C. [2]
D. 2

Question 3 – What is returned by: *tail [1]*

A. 1
B. [1]
C. []

Question 4 – What is the type of: *["hello"; "world]*

A. string
B. string list
C. (string, string)

Question 5 – What is returned by: *take 2 [5; 8; 5; 2]*

A. [2]
B. 58
C. [5; 2]
D. [5; 8]

Question 6 – What is returned by: *map (fun x -> x + 2) [4; 6; 7]*

A. [6; 8; 9]
B. [6; 6; 7]

Question 7 – What is returned by: *filter (fun x -> x = 2) [4; 2; 3; 2]*

A. false
B. [4; 3]
C. [2; 2; 2; 2]
D. [2; 2]

Question 8 – What does this function do?

A. Filters a list
B. Counts the length of a list
C. Maps a function over a list
D. Sums the elements of a list

```
let rec my_function : int list -> int =
  fun xs ->
    match xs with
      [] -> 0
    | (x :: rest) -> x + my_function rest ;;
```

Quiz 6

Question 1 – What is the name of the higher order function with type: *(‘a -> ‘b) -> ‘a list -> ‘b list*

A. map
B. filter
C. zip
D. fold

Question 2 – What is the name of the higher order function with type: *(‘a -> bool) -> ‘a list -> ‘a list*

A. map
B. filter
C. zip
D. fold

Question 3 – What is the name of the higher order function with type: *(‘a -> ‘b -> ‘a) -> ‘a -> ‘b list -> ‘a*

A. map
B. filter
C. zip
D. fold

Question 4 – What is the name of the higher order function with type: *‘a list -> ‘b list -> (‘a * ‘b) list*

A. map
B. filter
C. zip
D. fold

Question 5 – What will this do? *fun xs -> List.fold_left (fun x y -> x + 1) 0 xs*

A. add 1 to every element in a list
B. sum the elements in the list
C. make every element an even number
D. count the elements in the list

Answers

Quiz 1

1. B

2. B

3. A

4. B, C

5. A, B, C, D

6. B

7. C

8. B

9. C

10. B

11. C

12. B

13. D

Quiz 2

1. B

2. B

3. D

4. D

5. C

6. A

7. B

8. B

9. B

10. B

11. D

12. C

13. C

14. B

15. B

16. A

17. B

18. B

19. B

20. B

Quiz 3

1. B

2. B

3. C

4. B

5. D

6. B

7. B

Quiz 4

1. C

2. A

3. C

4. B

5. D

Quiz 5

1. B

2. B

3. C

4. B

5. D

6. A

7. D

8. D

Quiz 6

1. A

2. B

3. D

4. C

5. D

TOTAL: 62 points