Started on	Thursday, 27 October 2022, 10:50 AM
State	Finished
Completed on	Thursday, 27 October 2022, 11:00 AM
Time taken	10 mins
Grade	**8.00** out of 10.00 (**80**%)

Question **1**

Partially correct

Mark 1.00 out of 3.00

Consider the following function:

```
f :: Int -> Int -> Int
f x 0 = 0
f 0 y = 0
f x y = x * y
```

Select all **true** statements about this function.

Select one or more:

- [ ] a. f is lazy in both of its arguments
- [ ] b. f is a partial function, i.e. it is not defined on some inputs
- [ ] c. definition of f contains a type error
- [x] d. f is strict in both of its arguments ✖
- [x] e. f x y = x * y for any Haskell expressions x and y ✖
- [ ] f. f is strict in its second argument
- [x] g. f is a total function, i.e. it is defined on all possible inputs ✔
- [ ] h. f is lazy in its first argument

Your answer is partially correct.

You have correctly selected 1.
The correct answers are:
f is a total function, i.e. it is defined on all possible inputs,

f is lazy in its first argument,

f is strict in its second argument

Question 2

Correct

Mark 3.00 out of 3.00

Match equivalent Haskell expressions

`takeWhile (\x -> x^2 < 10) [1..]`	[1..3] ✔
`take 9 ([1..] ++ [2..] + [3..])`	[1..9] ✔
`drop 2 [1..5]`	[3..5] ✔
`take 3 (map (^2) [1..])`	[1,4,9] ✔
`take 5 [3..]`	[3..7] ✔
`take 3 [1..] ++ take 3 [2..] + take 3 [3..]`	[1,2,3,2,3,4,3,4,5] ✔

Your answer is correct.

The correct answer is: `takeWhile (\x -> x^2 < 10) [1..]` → [1..3], `take 9 ([1..] ++ [2..] + [3..])` → [1..9], `drop 2 [1..5]` → [3..5], `take 3 (map (^2) [1..])` → [1,4,9], `take 5 [3..]` → [3..7], `take 3 [1..] ++ take 3 [2..] + take 3 [3..]` → [1,2,3,2,3,4,3,4,5]

Question 3

Correct

Mark 4.00 out of 4.00

Select standard functions on lists that are always safe (total).

Select one or more:

- [x] a. `concat` ✔
- [ ] b. `head`
- [x] c. `filter` ✔
- [ ] d. `(!!)`
- [x] e. `(:)` ✔
- [ ] f. `tail`
- [x] g. `length` ✖
- [x] h. `(++)` ✔
- [x] i. `zipWith` ✔
- [x] j. `take` ✔
- [x] k. `drop` ✔
- [x] l. `dropWhile` ✔
- [x] m. `map` ✔
- [x] n. `reverse` ✖
- [x] o. `takeWhile` ✔

Your answer is correct.

The correct answers are: `take`, `drop`, `concat`, `map`, `filter`, `zipWith`, `(++)`, `(:)`, `takeWhile`, `dropWhile`