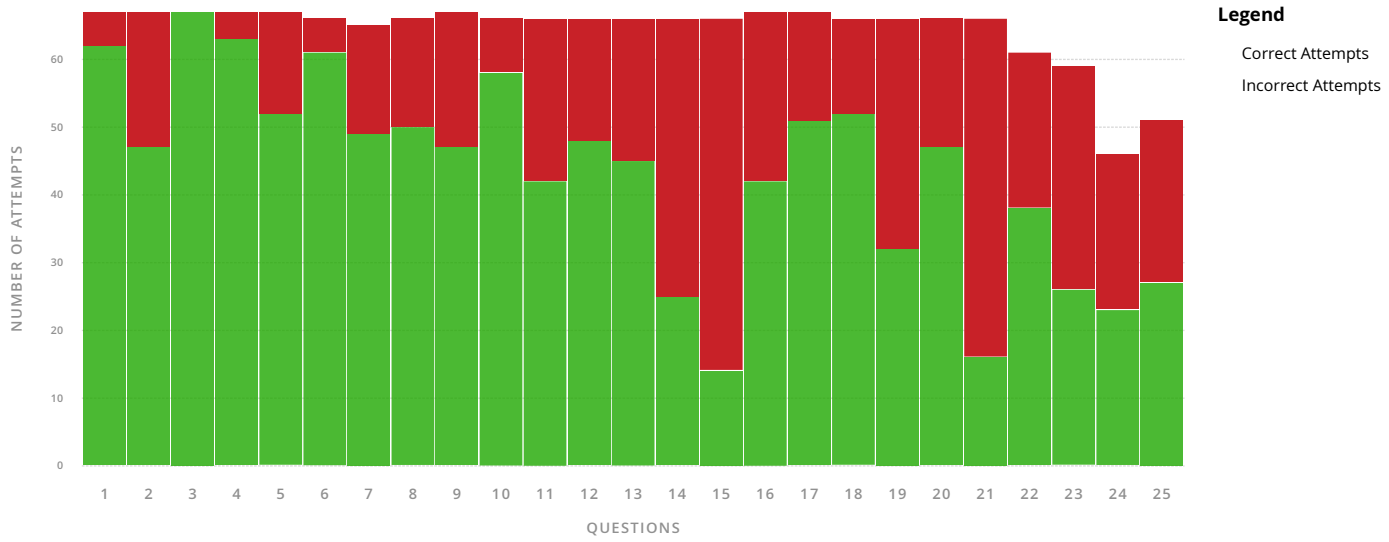
All attempts

Questions | Students

CLASS STATISTICS

17.13	Average	18	Median	3.932	Standard Deviation	9	Lowest	24	Highest

QUESTION STATISTICS



All Questions

Search by question

Qn.	Question Preview	Type	% Answered Correctly	Pt. Biserial

1	**Let Binding**
	Consider the Haskell expression where c1 is some constant.
	`let x =`
	`  let x=5 in x+c1`
	`in x+x`
	What result will be returned by the above expression under lazy evaluation?
		MCQ	92.54%	0.3564

2	**Let Binding**
	Consider the Haskell expression where c1 is some constant.
	`let x = c1 in`
	`let f y = x+(y+y) in`
	`  f ((f x)+2)`
	How many additions will be performed by above code fragment under lazy evaluation?
		MCQ	70.15%	0.2463

3	**Immutable Data Structure** What is an immutable data ?	MCQ	100%	0
4	What is **purely** functional programming? Choose the best answer below.	MCQ	94.03%	0.0246
	Recursion			

5	Consider the following recursive Haskell function.
	`foo x y =`
	`  if x<=0 then y+y`
	`  else (foo (x-1) (y*2))`
	What result will be returned by foo 3 c1, while c1 is some integer constant?
		MCQ	77.61%	0.1458

6	**Non-Termination**
	Consider the following recursive Haskell function.
	`foo x y =`
	`  if x=0 then y+y`
	`  else (foo (x-1) (y*2))`
	Does this function always terminates for all inputs, assuming the presence of infinite runtime resource?
		TOF	92.42%	0.2434

7	**Type Inference** Consider the following Haskell function. `foo g x = g x` Which of the following is the most general type of this function that would be inferred by Hindley-Milner type inference system used in the Haskell compiler?	MCQ	75.38%	0.3663

8	**Recursion**
	Consider the following recursive Haskell function.
	`foo x y =`
	`  if x<=0 then y+y`
	`  else (foo (x-1) (y*2))`
	Which of the following is NOT correct about this method definition?
		MCQ	75.76%	0.1337

9	**Recursion** Consider the following recursive function in Haskell. `foo g (x::Int) = g x (foo g x)` What is the most general type that could be inferred for this function?	MCQ	70.15%	0.3873
10	**Recursion** Consider the following recursive function in Haskell. `foo g (x::Int) = g x (foo g x)` What result would be returned by `(foo (\ n r -> n+r) c1)` where `c1` is a symbolic constant.	MCQ	87.88%	0.437

11	**Higher-Order Function**
	Consider the following simple higher-order function
	`foo xs g =`
	`  case xs of`
	`  []  -> (1,[])`
	`  y:ys ->`
	`  let (c,cs)= foo ys g in`
	`  if g y then (1+c,y:cs)`
	`  else (c,cs)`
	What value is returned by the expression `(fst (foo [-1,2,3,0] (\z -> z>0))` ?
		MCQ	63.64%	0.2893

12	**Higher-Order Functions**
	Consider the following higher-order function in Haskell.
	`gen_ho f xs n =`
	`  case xs of`
	`  [] -> []`
	`  y:ys -> (f y n):(gen_ho f xs (n+1))`
		MCQ	72.73%	0.1211

What is computed by the expression `(gen_ho f [c1,c2,c3] 1)`?

13	**Typing**
	Consider the following higher-order function in Haskell.
	`gen_ho f xs n =`
	`  case xs of`
	`  [] -> []`
	`  y:ys -> (f y n):(gen_ho f xs (n+1))`
	What would you do to this function definition to ensure that it will be correctly inferred by Haskell to have the type `((a->Int->b) -> [a] -> Int -> [b])` ?
		MCQ	68.18%	0.5045

14	**Typing**
	Consider the following higher-order function in Haskell.
	`gen_ho f xs n =`
	`  case xs of`
	`  [] -> []`
	`  y:ys -> (f y n):(gen_ho f xs (n+1))`
	What would you do to this function definition to ensure that it will be correctly inferred by Haskell to have the type `((a->Int->a) -> [a] -> Int -> [a])` ?
		MCQ	37.88%	0.2492

15	**Higher-Order Function**
	Consider the following higher-order function in Haskell.
	`gen_ho f xs n =`
	`  case xs of`
	`  [] -> []`
	`  y:ys -> (f y n):(gen_ho f xs (n+1))`
	Which of the following higher-order function can be used to implement this `gen_ho` function using just a single call to the selected higher-order function ?
		MCQ	21.21%	0.3714

16	**Higher-Order Function**
	Consider the following higher-order function in Haskell.
	`gen_ho f xs n =`
	`  case xs of`
	`  [] -> []`
	`  y:ys -> (f y n):(gen_ho f xs (n+1))`
	What would be returned by the call `(gen_ho f [1..] n)` ?
		MCQ	62.69%	0.1912

17	**Array** Consider the following Haskell expression `let a = array (1,10) ((1,1):[(i, if mod i 2==0 then a!(i-1) else i) | i <- [1..10]])` What is the value of `a!6` ?	MCQ	76.12%	0.3219
18	**Array** Consider the following Haskell expression `let a = array (1,10) [(i, if mod i 2==0 then a!(i-1) else i) | i <- [1..10]])` What is the value of `a!6` ?	MCQ	78.79%	0.3928
19	**Array** Consider the following Haskell expression `let a = array (1,10) [(i, if mod i 2==0 then a!(i-1) else i) | i <- [1..10]])` Which of the following is NOT true about this particular array?	MCQ	48.48%	0.1722

20	**List comprehension** Consider the list comprehension [g x y z | x in e1, y in e2, z in e3] Assume you have a filter condition (f x y) which you would like to add to the above list comprehension. Where would be the best place to put this filter?	MCQ	71.21%	0.1912
21	**List Comprehension** Consider the list comprehension: `[ (x,y) | x in e1, y in e2, z in e3]` Which of the statements below is NOT true?	MCQ	24.24%	-0.0288

22	**Higher-Order Function**
	Consider the following function
	`foo xs g =`
	`  case xs of`
	`  []  -> (1,[])`
	`  y:ys ->`
	`  let (c,cs)= foo ys g in`
	`  if g y then (1+c,y:cs)`
	`  else (c,cs)`
	One could re-implement the call `(foo xs g)` by an equivalent expression `(foldr e1 e2 xs)` which has the following type:
	`foldr :: Foldable t => (a -> b -> b) -> b -> t a -> b`
	Which of the following is/are correct?
		MRQ	62.3%	0.646

23	**Higher-Order Function**
	Consider the following function
	`foo xs g =`
	`  case xs of`
	`  []  -> (1,[])`
	`  y:ys ->`
	`  let (c,cs)= foo ys g in`
	`  if g y then (1+c,y:cs)`
	`  else (c,cs)`
	One could re-implement the call `(foo xs g)` by an equivalent expression `(foldl e2 e1 xs)`. Note that we are using `foldl` with he following type here:
	`foldl :: Foldable t => (b -> a -> b) -> b -> t a -> b`
	Which of the following is/are correct? This is a multiple response answer where all correct answer must be given.
		MRQ	44.07%	0.6321

24	**Type Annotation** Consider the following Haskell function. foo g = (g 1, g "hello") Is a user annotated type possible for this function? Justify your answer by explaining why it is possible or otherwise.	TOF	50%	0.1023
25	**Recursion** Consider the following recursive function in Haskell. `foo g (x::Int) = g x (foo g x)` It is <u>not</u> possible to implement the factorial function using this higher-order function since there isn't a base case. Is this statement true or false. Please qualify your answer.	TOF	52.94%	0.2536