Question **1**
Correct
Mark 1.00 out of 1.00
Flag question

Select the function that uses pattern guards correctly to implement the filter function:

a.
```
filter _ [] = []
filter p (x:xs)
  | p x -> x:filter p xs
  | else -> filter p xs
```

b.
```
filter _ [] = []
filter p (x:xs) =
  if p x then x:filter p xs
  otherwise filter p xs
```

c. ✔
```
filter _ [] = []
filter p (x:xs)
  | p x = x:filter p xs
  | otherwise = filter p xs
```

Your answer is correct.

Question **2**
Correct
Mark 1.00 out of 1.00
Flag question

Select the snippets that are valid Haskell code

a. ✔
```
inc :: Num a => a -> a
inc a = a + 1
```

b.
```
inc : Num a => a -> a
inc a = a + 1
```

c. ✔
```
len l = case l of
  [] -> 0
  (_:xs) -> 1 + len xs
```

d.
```
len l = case l of
  [] -> 0
  (_::xs) -> 1 + len xs
```

Your answer is correct.

Question **3**
Correct
Mark 1.00 out of 1.00
Flag question

Which of the following are examples of **valid** ways to create local definitions in Haskell?

a. `y * 2 where y = 5` ✔
b. `let y = 5 in y * 1` ✔
c. `local y = 5 in y * 2`
d. `y * 2 with y = 5`

Your answer is correct.

Question **4**
Correct
Mark 1.00 out of 1.00
Flag question

Given the following function definition:

```
f :: [Int] -> Int
f [1, 2] = 1
f [_, _] = 2
f [3, 4] = 3
```

the result of the following function call is:

```
f [3, 4]
```

Answer: 2 ✔

Question **5**
Partially correct
Mark 0.67 out of 1.00
Flag question

Which function describes best the each of the following list comprehensions?

`[ x^2 | x <- xs]`	map ✔
`[x | x <- xs, x `div` 3 == 2]`	filter ✔
`[ take 3 | x <- xs]`	take ✘

Your answer is partially correct.

Question **6**
Partially correct
Mark 0.50 out of 1.00
Flag question

Select all the **true** statements about the bottom value:

- a. In Haskell, Nothing is the bottom value
- b. The compiler won't compile (i.e. will show an error) programs that contain the bottom value
- c. In Haskell, undefined is the bottom value ✔
- d. The bottom value can be assigned to any type

Your answer is partially correct.
You have correctly selected 1.

Question **7**
Correct
Mark 1.00 out of 1.00
Flag question

Select all the **true** statements about type classes

- a. Type classes are used to abstract common behavior for various types (like Java interfaces) ✔
- b. All type class implementations for a data type must be in the module where the data is defined
- c. We can implement type classes defined by the standard library for our own types ✔
- d. Type classes are used to define classes, types that also have methods and private fields

Your answer is correct.

Question **8**
Incorrect
Mark 0.00 out of 1.00
Flag question

The following list comprehension:

```
[(x, y) | x <- ['a', 'b']; y <- [1, 2]]
```

- a. Fails to combile because the syntax is invalid
- b. Returns [('a', 1), ('b', 2)]
- c. Returns [('a',1),('a',2),('b',1),('b',2)] ✖
- d. Fails to combile because x and y have different types

Question **9**
Partially correct
Mark 0.25 out of 1.00
Flag question

Select the correct functions such that the definition of m3 below multiplies 3 numbers wrapped in Maybe

mul3 a b c = a * b * c

m3 a b c = mul3 <*> a <*> b <*> c

Your answer is partially correct.

Question **10**
Correct
Mark 1.00 out of 1.00
Flag question

Given the following code:

```
newtype All = All Bool
instance Semigroup All where
   (All a) <> (All b) = All (a && b)
instance Monoid All where
   mempty = All True
```

The result of the following expressions is:

`foldl (<>) mempty (map (\x -> All (x `mod` 2 == 0)) [1, 2, 3])` All False ✔

`foldl (<>) mempty (map (\x -> All (x >= 2)) [2, 3, 4])` All True ✔

Your answer is correct.