COMP105 Class Test 2 (Practice)

Worth 20% of total marks for the module
TIME ALLOWED : 35 minutes
Electronic devices are not permitted

Answer all questions. Answers should be filled in on the computer-readable answer sheet. Ensure that your student ID is filled out in both numeric and computer-readable form in the top right of the answer sheet.

Section A – Higher order functions

1. What is the result of the following query?

```
ghci> map (\ (_:y:_) -> y ) ["hello", "there"]
```

A. `"le"`
B. `"ht"`
C. `"eh"`
D. `"he"`
E. The query results in an error or an infinite loop.

2. What is the result of the following query?

```
ghci> filter (\ x -> x ) [True, False, True, False]
```

A. `[True,True]`
B. `[False,False]`
C. `[True,False,True,False]`
D. `[]`
E. The query results in an error or an infinite loop.

3. What is the result of the following query?

```
ghci> foldl (\ acc (a,b) -> a : acc ++ [b] ) [] [(1,2), (3,4)]
```

A. `[3,1,2,4]`
B. `[4,2,1,3]`
C. `[1,3,4,2]`
D. `[2,4,3,1]`
E. The query results in an error or an infinite loop.

4. What is the result of the following query?

```
ghci> scanr1 (\ _ acc -> acc ) [1,2,3,4]
```

A. `[1,1,1,1]`
B. `[4,4,4,4]`
C. `[1,2,3,4]`
D. `[4,3,2,1]`
E. The query results in an error or an infinite loop.

5. What is the result of the following query?

```
ghci> zipWith (!!) ["abc", "def", "geh"] [0,1,2]
```

A. `"adg"`
B. `"cfh"`
C. `"aeh"`
D. `"abc"`
E. The query results in an error or an infinite loop.

Section B – Types

6. What is the result of the following query?

```
ghci> (\ x y -> x + y + 2) 10 20
```

A. `4`
B. `12`
C. `22`
D. `32`
E. The query results in an error or an infinite loop.

7. What is the result of the following query?

```
ghci> take 2 . map (+1) . drop 2 $ [1..10]
```

A. `[1,2]`
B. `[5,6]`
C. `[3,4]`
D. `[4,5]`
E. The query results in an error or an infinite loop.

8. Consider the following function definition.

```
uncurry g (x, y) = g x y
```

What is the type of `uncurry`?

A. `(a -> b -> c) -> (a, b) -> c`

B. `(a, a) -> (a -> a -> a) -> a`

C. `(a, b) -> (a -> b -> c) -> c`

D. `(a -> a -> a) -> (a, a) -> a`

E. The function does not compile, and so it does not have a type.

9. Consider the following function definition.

```
is_two x = x == 2
```

What is the most general type for `is_two`?

A. `Int -> Bool`

B. `(Eq a, Num a) => a -> Bool`

C. `a -> Bool`

D. `(Eq a, Num a, Ord a) => a -> Bool`

E. `(Num a, Ord a) => a -> Bool`

10. Consider the following function definition.

```
p x y = x > 5 && show y == "hi"
```

What is the most general type for the function `p`?

A. `(Ord a, Show b) => a -> b -> Bool`

B. `(Ord a, Show b, Eq b) => a -> b -> Bool`

C. `(Ord a, Num a, Show b) => a -> b -> Bool`

D. `(Ord a, Num a, Show b, Eq b) => a -> b -> Bool`

E. `(Num a, Show b) => a -> b -> Bool`

Section C – Custom Types

11. Consider the following type definition.

```
data Weather = Hot | Warm | Cold | Freezing deriving (Show, Eq, Ord)
```

What is the result of the following query?

```
ghci> (Hot < Freezing, Warm >= Hot, Cold == Freezing)
```

A. `(False,False,False)`

B. `(True,True,False)`

C. `(True,False,False)`

D. `(True,True,True)`

E. The query results in an error or an infinite loop.

12. Using the same definition of `Weather` as used in Question 11, what is the result of the following query?

```
ghci> read "Hot" :: Weather
```

A. `True`

B. `"Hot"`

C. `[Hot]`

D. `Hot`

E. The query results in an error or an infinite loop.

13. Consider the following partially completed function, which is supposed to implement a safe version of the `!!` operator.

```
safe_index xs i
    | MISSING
    | otherwise = Nothing
```

Which of the following should `MISSING` be replaced with?

A. `i < length xs = Just (xs !! i)`

B. `i <= length xs = xs !! i`

C. `xs !! i /= Nothing = xs !! i`

D. `i < length xs = xs !! i`

E. `i <= length xs = Just (xs !! i)`

14. Which of the following will give an error if typed into ghci?

A. `[Left True, Right False, Left True, Right False]`

B. `[Left 'a', Right False, Left True, Right False]`

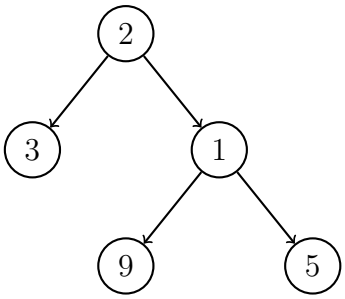
C. `[Left 'a', Right True, Left 'b', Right False]`

D. `[Left True, Right 'a', Left False, Right 'b']`

E. `[Left 'a', Right 'b', Left 'c', Right 'd']`

15. Consider the following custom data-tree type.

```
data DTree a = Leaf a | Branch a (DTree a) (DTree a) deriving (Show)
```



The tree above can be represented as a `DTree Int` in ghci like so

```
ghci> let tree = Branch 2 (Leaf 3) (Branch 1 (Leaf 9) (Leaf 5))
```

Suppose that we have loaded the following function into ghci.

```
tree_f (Leaf x) = [x]
tree_f (Branch x l r) = x : tree_f l ++ tree_f r
```

What is the result of the following query?

```
ghci> tree_f tree
```

A. `[3,9,5,1,2]`

B. `[3,2,9,1,5]`

C. `[5,1,9,2,3]`

D. `[2,1,9,5,3]`

E. `[2,3,1,9,5]`

Section D – IO and Models of Evaluation

16. The IO action `act` is defined as follows.

```
act :: IO Int
act = do
    x <- return 1
    y <- return 2
    z <- return 3
    return y
```

What is returned by the following query?

```
ghci> act
```

A. `1`

B. `2`

C. `IO 1`

D. `IO 2`

E. The query produces an error.

17. Using `act` as it is defined in Question 16, suppose that a user inputs the following query.

```
ghci> putStrLn (show (2*act))
```

What will be printed on the screen as a result of this query?

A. `2`

B. `4`

C. `"2"`

D. `"4"`

E. An error occurs.

18. What will be printed if the user types the following into ghci?

```
ghci> x = (head [], error "error")
ghci> y = fst x `div` (2 - 2)
ghci> y
```

A. `*** Exception: Prelude.head: empty list`

B. `*** Exception: divide by zero`

C. `*** Exception: error`

D. A type error will be printed.

E. `0`

19. Suppose that you are writing a function in Haskell, and that you need to use a fold. You know that the fold will always use the entire input list. Which of the following would be appropriate for this task?

A. `foldl`

B. `foldr`

C. `foldl'`

D. `filter`

E. `map`

20. Consider the following function.

```
mystery = 1 : zipWith (+) mystery mystery
```

What is the result of the following query?

```
ghci> take 5 mystery
```

A. `[1,3,9,27,81]`

B. `[1,2,3,4,5]`

C. `[1,2,4,8,16]`

D. The query produces an error.

E. The query enters an infinite loop.

Do not turn this over until the start of the test.

End