CS2104 Programming Language Concepts

Mid-Term Test

You have 50 minutes to complete the quiz. Use a B2 pencil to fill up the provided MCQ form.

After finishing, place the MCQ sheet on top of the question sheet and leave both on the table, when you exit the room.

Write your matrix card number in the box below.

This exam comprises 37 multiple choice questions with 5 possible answers each. In appropriate cases, the last option, None of the Above, should be chosen if the answer cannot be found in the first four choices. First two questions are for set identification purpose and will not be marked. Therefore, the last 35 questions will only be counted for marking.

The list of types of the library functions used in this questionnaire. Definition of some of the function is also provided.

```
(==) :: Eq a => a -> a -> Bool
(<) :: Ord a => a -> a -> Bool
(/) :: Fractional a => a -> a -> a
head :: [t] -> t
head (x:_) = x
tail :: [t] -> [t]
tail (x:xs) = xs
last :: [t] -> t
last ls = head (reverse ls)
init :: [t] -> [t]
init ls = reverse (tail (reverse ls))
sum  :: (Num a, Foldable t) => t a -> a
sum xs = foldl (\b a -> b + a) 0 xs
take :: Int -> [a] -> [a]
take 0 _ = []
take _ [] = []
take n (x:xs) = x:take (n-1) xs
trace :: String -> a -> a
foldl :: Foldable t => (b -> a -> b) -> b -> t a -> b
foldl f a [] = a
foldl f a (x:xs) = foldl f (f a x) x
zipWith :: (a -> b -> c) -> [a] -> [b] -> [c]
map :: (a -> b) -> [a] -> [b]
map f = foldl (\a x -> a ++ [f x]) []
filter :: (a -> Bool) -> [a] -> [a]
filter f = foldl (\a x -> if f x then a ++ [x] else a) []
zip :: [a] -> [b] -> [(a, b)]
zip xs ys = if xs == [] || ys == [] then [] else (head xs, head ys):zip (tail xs) (tail ys)
return :: Monad m => a -> m a
(>>=) :: Monad m => m a -> (a -> m b) -> m b
fmap :: Functor f => (a -> b) -> f a -> f b
($) :: (a -> b) -> a -> b
($) f a = f (a)              -- has the lowest precedence
(.) :: (b -> c) -> (a -> b) -> a -> c
(.) f g x = f (g x)          -- has the highest precedence
error :: [Char] -> a
show :: Show a => a -> String
```

Question 1: Please answer "E" for this question which is used to identify your question set.

1 [A] A

1 [B] B

1 [C] C

1 [D] D

1 [E] E

Question 2: Please answer "E" for this question which is used to identify your question set.

2 [A] A

2 [B] B

2 [C] C

2 [D] D

2 [E] E

Question 3:
What is the value of the following expression?

```
(\x y z -> z x y) (\p q -> p + q) 2 (\a b -> a) 3 5 :: Integer
```

3 [A] 2

3 [B] 7

3 [C] 8

3 [D] 3

3 [E] 5

Question 4:
Which one is the correct application of the following function?

```
f5 x1 x2 x3 = x1 + x2 + x3
```

4 [A] `f5 (0xC 1 0.00001)`

4 [B] `f5 (0xC, 1, 0.00001)`

4 [C] `((f5 0xC) 1) 0.00001`

4 [D] `f5 (0xC (1 0.00001))`

4 [E] None of the above

Question 5:
Which of the functions evaluates to 10 with the following argument?

```
[1,2,3,4,3,2,1]
```

5 [A] `(sum . take 3 . tail . tail)`

5 [B] `(sum . take 4 . head . tail)`

5 [C] `(sum . last 4)`

5 [D] Both a and b gives the result

5 [E] Both b and c gives the result

Question 6:
What is the value of f2 at the end?

```
f1 = 2
f2 =
  let f1 =
       let f2 = 3
       in
           f2 + f1
  in
      f1 + 1
```

6 [A] The function f2 evaluates to 3

6 [B] The function f2 evaluates to 6

6 [C] No output due to type mismatch

6 [D] No output due to infinite recursion

6 [E] The function f2 evaluates to 2

Question 7:
Consider the following Haskell fragment. What does it evaluate to?

```
let x=1
in let x=x*2
   in x*x
```

7 [A] 1

7 [B] 2

7 [C] 4

7 [D] 8

7 [E] None of the Above

Question 8:
Select appropriate code for ?? to calculate the best approximation of the square root of a number?

```
sqrt n =
  let aux x y e =
        if x - y <= e then
          x
        else
          let x' = ((x+y)/2) in
          aux x' (n/x') e
  in
      ??
```

8 [A] `aux 1 n 0.00001`

8 [B] `aux n (n*n) 0.0000001`

8 [C] `aux n (n/n) 0.0001`

8 [D] `aux n n 0.00001`

8 [E] None of the above

Question 9:
What is the correct result of the following code?

```
foo x y =
    if x <= 0 then y
    else 1+(foo (y/2) (x-2))
foo 10 20
```

9 [A] 5

9 [B] 6

9 [C] 10

9 [D] 11

9 [E] None of the above

Question 10:
Which of the following is correct about the foo method?

```
foo x y =
    if x <= 0 then y
    else 1+(foo (y/2) (x-2))
```

10 [A] This method uses tail recursion.

10 [B] This method does not have recursion.

10 [C] This method always terminates.

10 [D] This method has an accumulating parameter.

10 [E] This method is equivalent to integer division.

Question 11:
What is the type of `woo`?

```
woo a b c d = b a (woo c b (a==d))
```

11 [A] `Eq a => a -> (a -> a -> t -> t) -> a -> a -> t`

11 [B] `(Bool -> Bool) -> (Bool -> t) -> t) -> Bool -> Bool -> t`

11 [C] `Bool -> (Bool -> (Bool -> t) -> t) -> Bool -> Bool -> t`

11 [D] `Eq a => a -> (a -> a -> t -> t) -> a -> a -> t`

11 [E] `Eq a => Bool -> (Bool -> a -> t -> t) -> Bool -> a -> t`

Question 12:
What is the type of `s`?

```
s a b = a b (s a b)
```

12 [A] `(c->a->c) -> a -> c`

12 [B] `(a->c->a) -> a -> c`

12 [C] `(a->c->c) -> a -> c`

12 [D] `(a->c->c) -> c -> a`

12 [E] `(a->c->c) -> c -> c`

Question 13:
What is the most general type of f7?

```
f7 x1 x2 x3 =
   if x1 == x2 then
      x1 < x3
   else
      True
```

13 [A] `(a -> b -> c) -> Bool`

13 [B] `(Ord a, Num a, Show a) => a -> (a -> a -> Bool)`

13 [C] `(Ord a, Num a) => (a -> a -> a) -> Bool`

13 [D] `Ord a => a -> (a -> a -> Bool)`

13 [E] `Ord a => a -> (b -> a -> Bool)`

Question 14:
What is the most general type of the following function?

```
f8 = \x -> (x / x)
```

14 [A] `Double -> Double`

14 [B] `Int -> Double`

14 [C] `Num a => a -> a`

14 [D] `Num a, Fractional b => a -> b`

14 [E] `Fractional a => a -> a`

Question 15:
x is a fixed point of f if $fx = x$. Which is a fix point of `fib`?

```
fib x = if x <= 1 then x else (fib (x-1)) + (fib (x-2))
```

15 [A] 3

15 [B] 5

15 [C] 7

15 [D] 8

15 [E] 13

Question 16:
x is a fixed point of f if $fx = x$. How many fixed points does the function `(\x -> x + 1)` have?

16 [A] None

16 [B] One

16 [C] Two

16 [D] Infinity

16 [E] Depends on x

Question 17:
Which is the correct definition to generate an infinite list of fibonacci numbers, ifib? Note that ifib[0] = 0, ifib[1] = 1, and ifib[n] = ifib[n-1] + ifib[n-2]

```
zipWith f xs ys = map (\(x, y) -> f x y) $ zip xs ys
```

17 [A]
```
ifib = 0 : 1 : zipWith (+) (tail ifib) (tail ifib)
```

17 [B]
```
ifib = 0 : zipWith (+) ifib ifib
```

17 [C]
```
ifib = 0 : zipWith (+) ifib (tail ifib)
```

17 [D]
```
ifib = 0 : 1 : zipWith (+) ifib (tail ifib)
```

17 [E]
```
ifib = 0 : 1 : zipWith (+) ifib ifib
```

Question 18:
The following function `length` computes the length of a list. What is the value of the expression `length lfact`?

```
length xs =
   let aux [] len     = len
       aux (x:xs) len = aux xs (len+1)
   in  aux xs 0
lfact = 1:[i * j | (i, j) <- zip [2..] lfact]
```

18 [A] 0

18 [B] 1

18 [C] 2147483647

18 [D] Infinity

18 [E] Non-termination

Question 19:
Which is the result of evaluating the following expression?

```
(\ x -> let y = x / 0 in x + 1) 10
```

19 [A] Compile error

19 [B] Divide-by-zero exception

19 [C] Illegal operation exception

19 [D] Infinity

19 [E] The output is 11

Question 20:
Which of the following is not true about lazy evaluation in Haskell?

20 [A] Performance decreases due to performing needless calculations

20 [B] It provides the ability to define potentially infinite data structures

20 [C] It is the default evaluation mechanism in Haskell

20 [D] It delays evaluation of an expression until its value is needed

20 [E] Lazy evaluation can lead to reduction in memory footprint

Question 21:
Which line of the following code should be changed in order to perform a reverse and append on two lists, and how should it be changed? For example, rev [1,2,3] [4,5,6] is expected to be [6,5,4,3,2,1].

```
rev [] [] = []
rev [] ys = rev ys []
rev (x:xs) ys = x:rev xs ys
rev "abc" ['d','e','f']
```

21 [A] line 2. `rev [] (y:ys) = y:rev ys []`

21 [B] line 3. `rev (x:xs) ys = rev xs ys ++ [x]`

21 [C] line 4. `rev ['a','b','c'] ['d','e','f']`

21 [D] All of the above

21 [E] None of the above

Question 22:
Which expression should replace the last three lines to make it a tail recursive function?

```
isPal [] = True
isPal [_] = True
isPal (x:xs) =
    let l = last xs
        r = isPal (init xs)
    in  x == l && r
```

22 [A] `isPal (init xs)) && x == last xs`

22 [B] `(x == last xs && isPal (init xs))`

22 [C] `if isPal (init xs) then x == last xs else False`

22 [D] All of the above

22 [E] None of the above

Question 23:
Which one is the correct statement about fib1 and fib2?

```
fib1 n = let aux = 0:1:[i+j|(i,j) <- zip aux (tail aux)] in take n aux
fib2 n =
    let aux c a b = if c==0 then a else aux (c-1) (b:a) (head a + b)
    in  reverse $ aux n [0] 1
```

23 [A] fib1 is faster than fib2 in running time complexity

23 [B] fib1 is slower than fib2 in running time complexity

23 [C] fib1 and fib2 are almost same in running time complexity

23 [D] fib2 consumes more stack memory than fib1

23 [E] fib1 consumes more stack memory than fib2

Question 24:
What is the value of `(f5 [[0..i]|i<-[1..4]])`?

```
f5 = map $ sum . filter (\x -> mod x 2 == 0)
```

24 [A] [0,2,6]

24 [B] [2,2,6]

24 [C] [0,2,2,6]

24 [D] [10]

24 [E] [6]

Question 25:
Consider the following functions. Select the equivalent function.

```
foldl (\a x -> foldl (+) a x) 0
```

25 [A] `\x -> sum . map sum x`

25 [B] `\x -> sum $ sum x`

25 [C] `sum $ map sum`

25 [D] `sum . map sum`

25 [E] `\x -> map sum x`

Question 26:
What is the correct type of the following function `koo`?

```
koo b g f a x = if b x then a else koo b g f (g a x) (f x)
```

26 [A] `(t1 -> t) -> (t -> t1 -> t) -> (t1 -> t1) -> t -> t1 -> t`

26 [B] `(t1 -> Bool) -> (t -> t1 -> t) -> (t1 -> t1) -> t -> t1 -> t`

26 [C] `(t1 -> Bool) -> (t -> t1 -> t) -> (t1 -> Bool) -> t -> t1 -> t`

26 [D] `(t1 -> t) -> (t -> t1 -> t) -> (t -> t) -> t -> t1 -> t`

26 [E] `(t1 -> Bool) -> (t -> t -> t) -> (t1 -> t1) -> t -> t1 -> t`

Question 27:
Which one is the correct implementation of factorial function such that `fact 0 = 1` and `fact n = 1*2*3*..*(n-1)*n`

```
koo b g f a x = if b x then a else koo b g f (g a x) (f x)
```

27 [A] `fact n = koo (\x->x==n) (\a x->a*x) (\x->x-1) n 1`

27 [B] `fact n = koo (\x->x==1) (\a x->a*x) (\x->x-1) 0 n`

27 [C] `fact n = koo (\x->x==n) (\a x->a*x) (\x->x+1) 1 n`

27 [D] `fact n = koo (\x->x==n) (\a x->a*x) (\x->x+1) 0 n`

27 [E] `fact n = koo (\x->x==0) (\a x->a*x) (\x->x-1) 1 n`

Question 28:
Which is the correct statement about trf?

```
koo b g f a x = if b x then a else koo b g f (g a x) (f x)
trf n = fst $ koo (\x->x==n) (\(a,b) _->(b,a+b)) (\x->x+1) (0,1) 0
```

28 [A] `trf n = fib (n+1)` where $fib(0) = 0$, $fib(1) = 1$ and $fib(n) = fib(n-1)+fib(n-2)$.

28 [B] 'trf 10000' cannot be computed in resonable time due to exponential time complexity.

28 [C] 'trf 10000' may not be computed in resonable memory due to stack overflow.

28 [D] 'trf 10000' can be computed in resonable time and does not use the stack.

28 [E] b and c.

Question 29:
What is the value of `gen_ho (\x -> x-2) 6` as `gen_ho` is defined below?

```
gen_ho f n =
  if n==0 then []
  else (f n):(gen_ho f (n-2))
```

29 [A] [6,4,2,0]

29 [B] [4,2,0]

29 [C] [6,5,4,3,2,1,0]

29 [D] [6,4,2]

29 [E] [[6],[4],[2]]

Question 30:
What is the output of the expression `f [[2],[3]]`?

```
f m =
  do x1 <- m
     x2 <- filter (\x -> x > 5) x1
     x3 <- filter (\x -> x < 5) x1
     return (x2,x3)
```

30 [A] [[(2,2)],[(2,3)],[(3,2)],[(3,4)]]

30 [B] [(2,2),(2,3),(3,2),(3,4)]

30 [C] []

30 [D] [(2,3),(3,2)]

30 [E] None of the above

Question 31:
Which statement is not true about monads in Haskell?

31 [A] Monads provide composition of monadic computation

31 [B] Monads provide impure computation in a functional structure

31 [C] In Haskell, IO system is contructed using a monad

31 [D] `return` injects a value into a monad

31 [E] The bind operator is a higher order function

Question 32:
What is a possible implementation of `fmap`?

32 [A] `fmap f a = a >>= (\r -> f r)`

32 [B] `fmap f a = a >>= (\r -> f >>= (\q -> return q r))`

32 [C] `fmap f a = a >>= (\r -> f >>= (\q -> q r))`

32 [D] `fmap f a = a >>= (\r -> return $ f r)`

32 [E] `fmap f a = a >>= (\r -> (return . r) f)`

Question 33:
Suppose the following is the instanciation of Maybe from Applicative. Select the most appropriate expression for the missing part

```
instance Applicative Maybe where
    pure = Just
    Nothing <*> _ = Nothing
    (Just f) <*> something = error "Missing Part. To Be Implemented"
```

33 [A] `fmap f something`

33 [B] `f something`

33 [C] `return $ fmap f something`

33 [D] `return $ f something`

33 [E] None of the above

Question 34:
Which type of data structure is most usefull to build an infinite multiplication table?

34 [A] `[(Int,Int,Int)]`

34 [B] `Array Int (Int, Int)`

34 [C] `[[(Int,Int,Int)]]`

34 [D] `([Int,Int],[Int])`

34 [E] `Array (Int, Int) [(Int, Int, Int)]`

Question 35:
Which one is the correct implementation of the 'sieve' function to generate prime numbers?

```
sieve [] = []
sieve (x:xs) = ...
```

35 [A] `(sieve $ x:(filter (\y -> mod x y /= 0) xs))`

35 [B] `(sieve $ filter (\y -> mod x y /= 0) xs)`

35 [C] `[x] ++ (sieve $ filter (\y -> mod x y /= 0) xs)`

35 [D] `(sieve [y | y <-xs, (mod x y) /= 0])`

35 [E] `[x] ++ (sieve $ filter (\y -> mod y x /= 0) xs)`

Question 36:
Which one is not an equivalent to the following function? Note that f and g is said to be equivalent if and only if f x = g x for all x.

```
\f g xs -> [f x | x <- xs, g x]
```

36 [A] `\f g xs -> filter g $ map f xs`

36 [B] `\f g xs -> (map f . filter g) xs`

36 [C] `\f g xs -> map g $ filter f xs`

36 [D] `\f g xs -> (filter g . map f) xs`

36 [E] `\f g xs -> map f $ filter g xs`

Question 37:
What is the value of `take 5 trp`?

```
trp = [i*j*k | i <- [1..], j <- [2..], k <- [3..], mod k j == 0, mod j i == 0 ]
```

37 [A] [4,8,16,20,24]

37 [B] [8,12,16,20,24]

37 [C] [4,8,16,32,64]

37 [D] [8,16,20,24,28]

37 [E] None of the above