National Undergraduate Programme in Mathematical Sciences

Introduction to Programming

End-semester Examination, Semester I, 2024–2025

Date : November 25, 2024
Time : 0930–1200

Marks : 40
Weightage : 40%

This paper has two parts. Each Part A question is worth 2 marks. The marks for each Part B question are displayed next to the question. For Part A, provide short answers. For Part B, provide crisp and short programs. Syntax should more or less correspond to Haskell, but minor inaccuracies like wrong indentation will be ignored. You can assume standard library functions, but if you use something not taught in class, you must explain briefly what it does.

Part A

1. What is the value of `f [0,1,2,3,4]`, for `f` defined below? What is the type of `f`?

```
f = foldr (\a (x:xs) → (a:x):x:xs) [[]]
```

2. What is the value of `f 3 4`, for `f` defined below?

```
f 0 n = n+1
f m 0 = f (m-1) (m-1)
f m n = f (m-1) (f m (n-1))
```

3. Give four values that belong to the following data type.

```
data Tree = Node Bool [Tree]
```

4. What does the following expression do, and what is the number of steps taken, in terms of the length of `xs`?

```
foldl (\l a → reverse (a: reverse l)) [] xs
```

5. List all values in the data type `Maybe (Bool, Maybe (Maybe Bool))`.

6. Describe briefly what the `main` function below does,

```
main = replicateM 10 readLn >>= print . reverse . map not
```

Part B

1. Recall that **tails** is a library function which has the following behaviour.

```
tails [x1,x2, ... ,xn] = [[x1,x2, ... ,xn], [x2, ... ,xn], ... , [xn], []]
```

What is the number of steps taken by `f xs = map sum (tails xs)`, in terms of the length of `xs`? Define a function `g` with the same behaviour as `f`, but which takes $O(n)$ time on a list of length n. *(6 marks)*

2. Write a `main` function that reads 10 lines of user input, each an integer, and after reading each line, outputs the sum of all numbers read so far. *(10 marks)*

3. Consider the following data type declaration.

```
data Tree = Leaf Int | Fork Tree Tree
```

Define functions `inorder :: Tree → [Int]` and `mkTree :: [Int] → Tree` such that the following properties hold: *(12 marks)*

(a) If `xs` is of length n, `mkTree xs` is a tree of height $O(\log n)$.

(b) `inorder (mkTree xs) = xs`.

(c) `inorder t` runs in time proportional to the size of `t`, and `mkTree xs` runs in time proportional to the length of $|xs|$.

$T(k) = T(\ell) + T(r) + O(\ell)$

$= T(O(k) + \ell)$

$= O(k)$

$\exists c$

c

$\forall m < k :$

$T(m) = cm.$

Scanned with OKEN Scanner