I I T G O A

CS300 Programming Language Paradigms— Midsem

12 October 2022

Time: **2 hr**
Total marks: **30**
Total Pages: **3**

Answer **all** *questions. Make your answers short and precise.*

1. (2 marks) What are the types of the following values? If the expression is invalid, briefly state why.

   (a) `(+1) $ (0 <)`

   (b) `subst f g x = f x (g x)`

2. (2 marks) What are the values of these expressions?

   (a) `foldl (-) 0 [1,2,3]`

   (b) `foldr (-) 0 [1,2,3]`

3. (2 marks) What is the most general type of the following function? Decsribe in brief what the function does with an example.

```
f [] y = y
f _ [] = []
f (x:xs) (y:ys) = f xs ys
```

4. (3 marks) Write a function `pairswap :: [a] -> [a]` that swaps the elements of a list pair-wise, i.e., the first and second elements are swapped, the third and fourth are swapped and so on. Assume the list has an even number of elements and is possibly empty.

```
>pairswap [1,2,3,4,5,6]
 [2,1,4,3,6,5]
```

5. (3 marks) A list of strings is *connected* if each string in the list (except the first) is obtained from the previous one by changing the character in *exactly one* position.

   Define a function `connected :: [String] -> Bool` that checks whether the input list of strings is *connected*.

For example,

```
>connected []
 True
>connected [''aa'',''ab'',''ba'']
 False
>connected [''aa'',''ab'',''bb'',''ba'']
 True
>connected [''ab'',''ab'']
 False
```

6. (3 marks) Write a function `combine :: [a] -> [[a]]` that groups consecutive duplicates into sublists. For example,

   ```
   >combine [2,2,4,4,4,6]
    [[2,2],[4,4,4],[6]]
   ```

7. (3 marks) Using `foldr`, write a function `separate :: [Char] -> ([Char],[Char])` that takes a string `s` and returns a 2-tuple with the digits and non-digits in the string `s` separated, with the initial order maintained. For example,

   ```
   >separate ''July 4, 1994''
    (''41994'',''July , '')
   ```

   Below is the partial implementation of `separate` using foldr. You need to write the folding function `f`.

   ```
   separate s = foldr f ([],[]) s
   ```

8. (4 marks) Write a function `scanl :: (a -> b -> a) -> a -> [b] -> [a]` that is the verbose version of the function `foldl`. It returns the list of successive values obtained by applying `foldl`. Some examples are given below.

   ```
   >scanl (\ x y -> 2*x + y) 4 [1,2,3]
    [4,9,20,43]
   >scanl max 5 [1,3,4,6,7,8]
    [5,5,5,5,6,7,8]
   ```

9. (4 marks) Using list comprehension, define a function `partitioned :: [Int] -> Bool` that returns `True` if there is an element n of the list such that:

   - for each element m occurring before n in the list, $m \leq n$, and
   - for each element m occurring after n in the list, $m > n$.

For example,

```
>partitioned []
 False
>partitioned [22]
 True
>partitioned [19,17,18,7]
 False
>partitioned [7,18,17,19]
 True
>partitioned [19,13,16,15,19,25,22]
 True
```

10. (4 marks) A *segment* of a list `xs` is a selection of adjacent elements of `xs`.

    Define a function `segment :: [a] ->[[a]]` that takes a finite list `xs` as its argument and returns the list of all the segments of `xs`.

    For example,

```
>segment []
 [[]]
>segment [1,2,3]
 [[1,2,3],[1,2],[2,3],[1],[2],[3]]
```