Programming Languages
24 August 2022

Rules

Time at disposal: 4 h.

Read carefully the following questions. For every questions, provide an answer in English. Please remember to justify why the results hold by writing the logical flow that lead you to the answers.

Note that every question can have one or more points. Points can be answered individually (no need to answer all the points of a question). To pass the exam, a perfect answer for all questions is not needed.

It is recommended to start with the questions that you believe are easy/quick, addressing the easier/quicker points and then move to the questions that you deem more elaborate. It is important to avoid prioritizing only the questions on the theory part (i.e., question 1-4) or the questions on Haskell (i.e., question 5-8). A mix of answers should be present to demonstrate a sufficient knowledge of both topics. Clearly, the more answers are provided correctly, the higher is the chance to pass and the higher will be the final grade obtained.

Exercises

1. Please provide an answer to the following items.

   (a) Define in your own words the notion of overriding. Give an example in pseudocode of overriding.

   (b) Define in your own words the notion of overloading. Give an example in pseudocode of overloading.

   (c) Consider the following pseudocode in a language Java-like where if a class A extends B then A is a subtype of B.

   ```
   class A { ... }
   class B extends A { ... }
   class C extends B { ... }

   class F {
   ```

```
  B fie (B x) {...}
}
```

Now consider the following classes.

```
class Q1 extends F {
  A fie (A p) {...}
}

class Q2 extends F {
  C fie (C p) {...}
}

class Q3 extends F {
  A fie (C p) {...}
}

class Q4 extends F {
  C fie (A p) {...}
}
```

For each of these 4 classes, state if they present or not a threat to type security (i.e., if it is possible to have type errors at runtime by allowing writing such a class)? Motivate your answer.

2. Please provide an answer to the following items.

   (a) Describe what an Activation Record is and what kind of information it contains.

   (b) In the context of Memory Management, describe what the Stack is and why it is useful.

   (c) Describe the technique of Display to implement the static scoping. What advantages does it bring in comparison with the usage of follow the static chain pointer?

3. Please provide an answer to the following items.

   (a) Name at least two iterative commands and describe what they do.

   (b) Name at least two commands for explicit sequence control and describe what they do.

   (c) Describe why the usage of the `goto` command is discouraged.

   (d) What is a *tail recursive* function? Write the pseudocode of a tail recursive function.

4. Please provide an answer to the following items.

   (a) Consider the following pseudo code of a language using static scoping and passing parameters by name.

```
int i = 1;
int [] A = new int [5];
void fie (name int x, name int y) {
  int i = 3;
```

```
      x = x+1;
      y = 1;
      A[i] = 3;
      A[A[i]] = 4;
      }
    for (j = 0; j <= 4; j += 1) {
      A[j] = 0;
    }
    fie(i, A[i]);
    write(A[1]);
    write(A[2]);
    write(A[3]);
    write(A[i]);
```

What does it print? Motivate your answer.

(b) What does the previous program print if dynamic scoping is used instead of static scoping?

(c) What does the previous program print if static scoping and call by value is used?

(d) What happens instead if static scoping and call by reference is used?

5. Please provide an answer to the following items.

   (a) Write a function `foo` in Haskell using pattern matching that given a list of triples of 3 elements returns a triple of list containing respectively: i) the first elements of the tuples, the second elements of the tuple, and the third elements of the tuple.

   For example.

   ```
   foo [(0,1,True),(2,3,False),(4,5, True)] = ([0,2,4],[1,3,5],[True,False,True])
   ```

   Write the type signature of all the functions that you define.

   (b) Describe the purpose of the keyword `where` in Haskell. Define a simple expression in which the `where` keyword is used.

6. Briefly describe how you parse the initial configuration text from a file in your Onitama project to produce your internal representation.

7. A lot of techniques for the compression of images are based on a tree data structure called "Quad Tree". Assume that the image is square and the size of the square is a power of 2. If the image is homogeneous (same color) it is encoded, regardless of its dimension, as a leaf containing its color (see Figure 1 for a graphical representation).

   If the image is not homogeneous, then it is encoded as a node whose child encode i) the upper left square, ii) the upper right square, iii) the bottom right square, and iv) the bottom left square using the data type

```
data (Eq a, Show a, Num a) => QT a = C a | Q (QT a ) (QT a ) (QT a ) (QT a )
  deriving (Eq, Show)
```

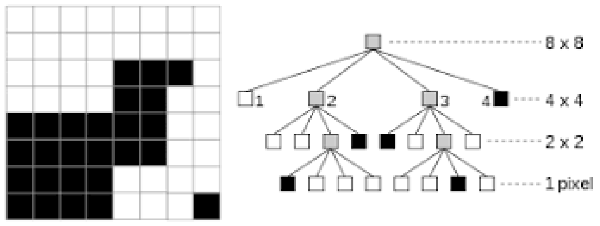

Figure 1: Graphical representation of a Quad Tree.

Write a Haskell function `myadd` that given a non empty list of QuadTrees `ls` returns a QuadTrees of pixels such that its pixel have as color the sum of colors of all the pixels in the same position of the trees in `ls`.

For example. Suppose that

```
z = C 0
u = C 1
q = (Q z u u u)
f = myadd [u, u, q]
```

Then `f` will output `Q (C 2) (C 3) (C 3) (C 3)`.

Write the type signature of all the functions that you define.
Motivate your choices briefly describing your code.

8. Please provide an answer to the following items.

   (a) Consider the following Haskell program

   ```
   myfunction = do
     f <- (5*)
     (return f) . (\x -> x+1)
   ```

   Give the type signature of `myfunction`. What Monad is used in this case?

   (b) Unfold the do syntactic sugar notation of `myfunction`.

   (c) Consider the standard function mapM that can be defined on list as follows.

   ```
   mapM :: Monad m => (a -> m b) -> [a] -> m [b]
   mapM _ [] = return []
   mapM f (x:xs) =  do
     y <- f x
     ys <- mapM f xs
     return (y:ys)
   ```

   The list is an instance of Monad where

```
return x = [x]
xs >>= f = concat (map f xs)
```

where concat is the function that concatenates lists that can be defined as follows.

```
concat [] = []
concat (x:xs) = x ++ (concat xs)
```

Show the steps that Haskell takes to compute the following expression.

```
mapM (\x -> x) [[0,1],[2]]
```