Programming Languages
28 June 2021

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, addressing the easier 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.

At the end of the exam, remember to submit the assignment to the PeerGrade system and in system DE-Digital Exam in due time. The same pdf should be submitted to both systems. Remember to avoid reporting your personal name and surname in the text submitted (the elaborate should be anonymous).

Exercises

1. Please provide an answer to the following questions.

   (a) Describe in your own words what is an exception and what is the difference between an exception and an error.

   (b) Consider the following pseudo code of a language with exceptions.

```
void f() throws X {
  throw new X();
}
```

```
void g (int sw) throws X {
  if (sw == 0) { f(); }

  try { f(); }
  catch (X e) { write("in_g"); }
}

...

try { g(1); }
catch (X e)  { write("in_main"); }
```

What does this code print when executed? Please explain why.

(c) Please describe how exceptions can "naively" be implemented by using the the stack and activation records.

(d) Describe one limitation of implementing exceptions by using only the the stack and activation records.

2. Please provide an answer to the following questions.

(a) List at least 3 imperative languages defined before the 70s, briefly explaining why they are still remembered today (e.g., did they introduce new features, was it the first language to do X, ...)

(b) List the name a functional programming language defined in the late 50s. Named at least one feature it introduced to the programming language scene.

(c) Why is the Simula language famous for?

(d) Why is the Prolog language famous for?

3. Please provide an answer to the following questions.

(a) In the context of Object orientation, define what we mean with the term inheritance. Why is it useful?

(b) What are the differences of single and multiple inheritance?

(c) Name a problem that can emerge when dealing with multiple inheritance.

(d) Describe how single inheritance can be implemented using linked lists.

4. Please provide an answer to the following questions.

(a) Consider the following pseudo code of a language using static scoping, call by reference, and evaluating expressions from left to right.

```
{
 int x = 0;
 int A(reference int y) {
   int x=2;
```

```
   y=y+1;
   return B(y)+x;
}
 int B(reference int y){
   int C(reference int y){
     int x = 3;
     return A(y)+x+y;
   }
   if (y==1) return C(x)+y;
     else return x+y;
 }
 write (A(x));
 write (x);
}
```

What does it print? Motivate your answer.

(b) What does the previous program print if dynamic scoping is used instead of static scoping? Motivate your answer and show how the stack looks like immediately after the invocation of B for the second time.

(c) What does the previous program print if static scoping, call by reference are used but expressions are evaluated from right to left?

5. Please provide an answer to the following items.

(a) Describe with your own words what is a class and what is an instance in Haskell.

(b) Consider the following data type that represent the status of a semaphore with its color (red, yellow, or green) and with the intensity of the light emitted represented as a Float.

```
data Semaphore = Red Float | Yellow Float | Green Float
```

Assume that `s1` of type Semaphore is equal to `s2` if they have the same color no matter the intensity of the light. For example

- `Red 0.1` is equal to `Red 0.1`
- `Yellow 0.5` is equal to `Yellow 0`
- `Green 1` is not equal to `Yellow 1`

Define the instance of Semaphore that allows to check for the equality of two values of type Semaphore.

(c) Consider the possibility of defining Semaphore as follows.

```
data Semaphore = Red Float | Yellow Float | Green Float deriving Eq
```

Would the equality operator defined using the keyword `deriving` implement the definition of equality stated in the second point above? If not, please explain why.

6. What function have you used to generate random numbers or moves in the Onitama project? Give an example of how a developer can generate two random values in Haskell by using the function you used in your Onitama project.

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

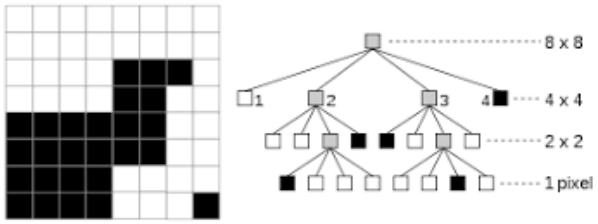

Figure 1: Graphical representation of a Quad Tree.

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 QT a = C a | Q (QT a ) (QT a ) (QT a ) (QT a )
```

Write a Haskell function `myInsert` that given two QuadTrees (`qt` and `qf`) of images and a boolean QuadTrees `qb`, builds the QuadTree keeping the pixels of `qt` for which the corresponding pixel of `qb` are True, otherwise keeping the pixels of `qf` (when the corresponding pixels of `qb` are false).

For example. Suppose that

```
z = C 0
u = C 1
qb = (Q (C True) (C False) (C True) (C False)
qf = Q z u u u
```

Then the function `myInsert (C 4) qf qb` will output `Q (C 4) (C 1) (C 4) (C 1)`.

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 questions.

   (a) The list type is an instance of the 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)
```

Consider the following code

```
mymess x = do
  [x,x]
  y <- ['a']
  return (y,y)
```

Give the type signature of mymess.

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

(c) What is the result of the evaluation of the expression `mymess 1`. Show the main steps that Haskell is performing to arrive at the result.