Programming Languages
20 August 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 items 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 system DE-Digital Exam in due time.

Exercises

1. Please provide an answer to the following items.
   (a) Describe the concept of overriding.
   (b) Describe the concept of overloading.
   (c) What does it means that a method is virtual? Has Java virtual methods?
   (d) Consider the following C++-like pseudo-code language allowing the usage of virtual methods by using the `virtual` keyword.

```
class Base {
public:
    void NonVirtual() {
        write("Not virtual");
    }
    virtual void Virtual() {
        write("Not virtual");
    }
}

class Derived : public Base {
public:
    void NonVirtual() {
        write("Derived not virtual");
    }
    void Virtual() {
        write("Derived virtual");
    }
};

int main() {
    Base a = new Base();
    Base b = new Derived();

    a.NonVirtual();
    a.Virtual();
    b.NonVirtual();
    b.Virtual();
}
```

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

2. Please provide an answer to the following items.

   (a) Describe what an activation record is.

   (b) Describe what a dynamic chain pointer is.

   (c) What do we mean with static memory management?

   (d) In what conditions a static memory management is not enough to implement an interpreter or compiler for a language? Explain why.

3. Please provide an answer to the following items.

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

   (b) What is a *tail recursive* function?

   (c) Provide the pseudo-code of a tail recursive function that computes the factorial of a positive integer (i.e., $n! = n * (n-1) * (n-2) * \ldots * 1$).

   (d) Explain why a tail recursion function can lead to performance gains compared to a non tail-recursive function.

4. Please provide an answer to the following items.

(a) Describe the notion of closure. What is it used for?

(b) Describe what *deep binding* and *shallow binding* are.

(c) With static scoping, can we use deep biding, shallow binding, or both? Why?

(d) The following program is written using a pseudo-code. Assume that dynamic scoping is used. What does the following program write if shallow binding is used? Motivate the answer.

```
{ int x = 0;
  int f(int y){
    return x+y;
  }
  int g (int h(int b)){
    int x = 2;
    return h(1) + x;
  }
  { int x = 7;
    int z = g(f);
    write(z)
  }
}
```

(e) What does the previous program write if deep binding is used instead of shallow binding? Motivate the answer.

5. Please provide an answer to the following items.

   (a) What do we mean when we say that the functions in Haskell are *curried*?

   (b) Define the function `flip` that takes a function and returns a function that is like the original function, but with the first two arguments flipped. Write the type signature of `flip`.
   As an example `(flip (++)) "a" "b"` will output `"ba"`.

   (c) Describe in your own words how the *type inference* mechanism works.

   (d) Consider the function `f x y z = (x z) (y z)`. What is the type of `f`. Motivate your answer.

6. Describe how you defined the configuration of an Onitama game using data types in your Haskell project. Write a possible data type definition for the game state and one example of value for that data type representing an Onitama game configuration.

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

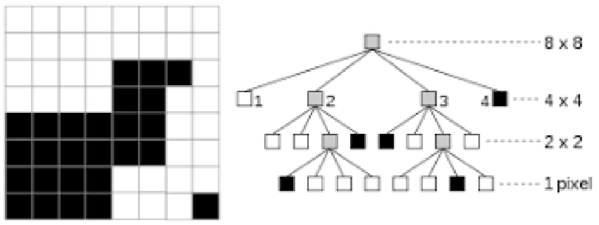

Figure 1: Graphical representation of a Quad Tree.

Write a Haskell function `myCount` that given a color, the size of the image (always a power of 2), and one QuadTrees computes the number of pixels of the given color.

For example. Suppose that

```
z = C "red"
u = C "blue"
q = (Q z u u (Q z u u u))
```

Then the function `myCount "blue" 4 q` will output `11` while `myCount "red" 8 q` will output `20`.

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

8. The *state monad* is defined as follows.

```
newtype State s a = State { runState :: s -> (a, s) }

instance Monad (State s) where
  return x = State $ \s -> (x, s)
  (State h) >>= f = State $ \s -> let (a, newState) = h s
                                      (State g) = f a
                                  in g newState
```

Assume the following definitions:

```
pop :: State [Int] Int
pop = State (\(x:xs) -> (x,xs))

push :: Int -> State [Int] ()
push a = State (\xs -> ( (), a:xs))

mymess = do
  push 2
  pop
```

Please provide an answer to the following items.

(a) 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 `runState mymess [1]`. Show the main steps that Haskell is performing to arrive at the result.