CMPS112 Winter 2016, Example Questions

1. (a) What is a pure function? Describe in one sentence and give one example.
   (b) What is an impure function? Describe in one sentence and give one example.
   (c) What is the difference between lazy and eager evaluation?
   (d) What is a type class in Haskell? Describe in one sentence and give one example.

2. Fill in the blanks in the following examples such that it produces the desired output. (10pt)

   ```haskell
   let a = ["very","much","so"]
   let b = ["currency","coin","crypto"]
   ```

   ```haskell
   [ x | x <- a ]  --> ["very","much","so"]
   [ _ | x <- b ]  --> [8, 4, 6]
   [ _ | x <- a ]  --> "vms"
   [ x | x <- a, _ ]  --> ["very","much"]
   [ _ | x <- [1..5] ]  --> [1,4,9,16,25]
   [(x,y) | x <- [1..3], y <- [1..3], _ ]  --> [(1,1),(1,2),(1,3),(2,1),(2,2),(3,1)]
   [ x ++ " " ++ y | x <- a, y <- b, _ ]  --> ["very currency","so crypto"]
   [ x | _ ]  --> "verymuchso"
   ```
3. (a) What is unification? *Describe in one sentence and give one example.*

(b) What is backtracking?

4. Assuming that the `append(A, B, Res)` predicate is true if both A and B are lists and Res is the result from concatenating these lists, write the output for the following Prolog queries right next to each query:

```
?- append([], [], []).

?- append([1], [2], [1, 2]).

?- append([], [], X).

?- append(X, [2], [1, 2]).

?- A = [1], B = [2|A], C = [3|B].

?- X = [R], append(X, Y, [1, 2, 3]).

?- append([1|Q], [], [1, 4]), Q = [N], D is N - 1.
```
5. (a) What is a closure? Describe in one sentence and give one example.
(b) What is a loop invariant? Describe in one sentence and give one example.

6. Write a isSorted function in Haskell which returns true if the given array of numbers is sorted in ascending order. (Hint: Use recursion and pattern matching).

\[
isSorted :: (\text{Ord } a) \Rightarrow [a] \rightarrow \text{Bool}
\]

-- Examples:
-- isSorted [1,2,3] --> True
-- isSorted [2,3,1] --> False

7. Write the isSorted predicate in Prolog by using recursion and unification.

% Examples: ?- isSorted([1,2,3]).
% yes.
% ?- isSorted([2,3,1]).
% no.
8. Add a post condition and two loop invariants to the following IsSorted method in Dafny.

```dafny
method IsSorted(arr: array<int>) returns (res: bool)
  requires arr != null;

  ensures

  { if (arr.Length < 2) { return true; }
    var n := 1;
    while (n < arr.Length)
      invariant
      invariant
        { if (arr[n - 1] > arr[n]) {
          return false;
        }
        n := n + 1;
      }
      return true;
  }

method TestIsSorted()
{ var arr := new int[3];
  arr[0] := 21;
  arr[1] := 21;
  var r := IsSorted(arr);
  assert r;
}
```
9. For this question, we are going to define a few different geometric shapes.

    type Point = (Double, Double)
    data Square = Square Point Double -- topLeft sideLength
    data Rectangle = Rectangle Point Double Double -- topLeft width height
    data Circle = Circle Point Double -- center radius

First, we want to be able to check whether a point is contained by a shape. Therefore, we add a special `Contains` type class.

    class Contains a where
        containsPt :: a -> Point -> Bool

The following instance definition ensures that we can test containment for `Squares`:

    instance Contains Square where
        containsPt (Square (x, y) l) (x', y') = x <= x' && x' <= x + l &&
                                                  y <= y' && y' <= y + l

(a) Make `Rectangle` a member of the `Contains` type class with an instance definition that includes implementations of the functions above.

    instance Contains Rectangle where
        containsPt

(b) Make `Circle` a member of the `Contains` type class with an instance definition that includes implementations of the functions above.

    instance Contains Circle where
        containsPt

10. Additionally, we create a type class for shapes. For every shape, we can test whether a given point is in the shape and we can also get its circumference and area.

    class (Contains a) => Shape a where
        circumference :: a -> Double
        area :: a -> Double

(a) Make `Square` a member of the `Shape` type class with an instance definition that includes implementations of the functions above.

    instance Shape Square where
        circumference

        area
(b) Make `Rectangle` a member of the `Shape` type class with an instance definition that includes implementations of the functions above.

```haskell
instance Shape Rectangle where
    circumference
    area
```

(c) Make `Circle` a member of the `Shape` type class with an instance definition that includes implementations of the functions above.

```haskell
instance Shape Circle where
    circumference
    area
```

11. There are different ways of combining shapes, like union, intersection or subtracting one shape from another. If we represent these operations as algebraic datatypes, we can check for point containment in terms of the two involved shapes. Complete the following instance definition by implementing `containsPt` for all three operations.

```haskell
data ShapeCombination a b = Union a b | Intersect a b | Diff a b

instance (Contains a, Contains b) => Contains (ShapeCombination a b) where
```

12. What is the result of the following Haskell code fragments?

(a) do a <- Just 23
    guard $ a < 10
    b <- Just 42
    return $ a + b

(b) do a <- Just 23
    guard $ a < 30
    b <- Just 42
    return $ a + b

(c) do a <- Nothing
    b <- Just 42
    return $ b + b