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"]

[ 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 the difference between prototype-based inheritance and class-based inheritance?

6. The following JavaScript code creates new rectangle objects with methods `getHeight` and `getWidth`.

```javascript
function Rectangle(h, w) {
    this.h = h;
    this.w = w;
}
Rectangle.prototype.getHeight = function() { return this.h; }
Rectangle.prototype.getWidth = function() { return this.w; }
```

(a) Implement the functions `getArea` and `getPerimeter` for all rectangles by completing the following code fragment, using the functions `getHeight` and `getWidth`.

```javascript
Rectangle.prototype.getArea = function() {
    // Example:
    // var rect = new Rectangle(5,6);
    // rect.getArea() // --> 30
    // rect.getPerimeter() // --> 22
}

Rectangle.prototype.getPerimeter = function() {
    // Example:
    // var rect = new Rectangle(5,6);
    // rect.getArea() // --> 15
    // rect.getPerimeter() // --> 20
}
```

(b) Suppose we also want to create squares by using a single length for both width and height. Squares are also rectangles, so we set the prototype accordingly.

```javascript
function Square(a) {
    this.a = a;
}
Square.prototype = Object.create(Rectangle.prototype);
```

Unfortunately, `new Square(5).getArea()` and `new Square(3).getPerimeter()` both result in an exception with the implementation above. Make the two examples work by implementing `getWidth()` and `getHeight()` for the Square prototype.

```javascript
Square.prototype.getWidth = function() {
    // Example:
    // var mySquare = new Square(5);
    // mySquare.getArea() // --> 25
    // mySquare.getPerimeter() // --> 20
}

Square.prototype.getHeight = function() {
    // Example:
    // var mySquare = new Square(5);
    // mySquare.getArea() // --> 25
    // mySquare.getPerimeter() // --> 20
}
7. Consider the following JavaScript function which returns true if the given array of numbers is sorted in ascending order.

```javascript
function isSorted(arr) {
    for (var i = 0; i < arr.length - 1; i++) {
        if (arr[i] > arr[i+1]) return false;
    }
    return true;
}
```

(a) Write the `isSorted` function in Haskell by using recursion and pattern matching.

```
isSorted :: (Ord a) => [a] -> Bool
```

**Examples:**

```
-- isSorted [1,2,3] --> True
-- isSorted [2,3,1] --> False
```

(b) Write the `isSorted` predicate in Prolog by using recursion and unification.

```
% Examples:  ?- isSorted([1,2,3]).
%           yes.
% ?- isSorted([2,3,1]).
%           no.
```