1. You manage two junior programmers and overhear the following conversation:

   A: "I don't know why anyone needs a language other than Java, it provides clean thread-based parallel programming"

   B: "Maybe, but I write my parallel programs in a functional programming language because they are then embarrassingly parallel"

   Discuss the correctness of these statements and the extent to which they cover the range of languages for parallel programming.

   Java only provides green-threads. While these provide concurrency, they do not provide parallelism (user-level threading). Programs written in a functional programming language *are* embarassingly parallel. However, this sets the limits of parallelism – and modern functional languages are not even close to these limits. In many cases, running small functions on different cores or threads is more overhead than running it serially. Therefore, many implementations do not exploit this massive possibility for parallelism.

   OCaml 5 has added multicore support by default. However, the speedup observed when compared to OCaml 4 is small. While OCaml 5 *is* faster on many workloads, the difference in speed is not significant on most workloads.

2. What is the difference between internal and external iteration?

   External iteration is normal iterating over a loop. Internal iteration is nested loops where each internal loop is executed on a different core. For example:

   External Iteration:

```
for (int i = 0; i < n; i++){
        // body
}
```

   Internal Iteration:

```
for (int core = 0; core < NUMCORES; core++){
        for (int i = 0; i < n / NUMCORES; i++){
                // body
        }
}
```

3. Scripting languages and dynamically typed languages are identical; discuss.

   Scripting Languages are a type of languages which are intended to be used for writing short programs which are used once (or a small number of times) and subsequently thrown away. While dynamically typed languages are languages which perform tyepchecking at runtime.

   Scripting languages are often dynamically typed, there are dynamically typed languages (such as python) which are not scripting languages.

4. Discuss the notion of "class" in relation to JavaScript.

5. Explain what is meant by a monad in a programming language, giving the two fundamental operations of a monad along with their types.

6. Consider the use of a monad for input-output. For the purposes of this question, take the IO monad as including two operations `readint` and `writeint` which respectively read integers from `stdin` and write integers to `stdout`. Give the types of these operators.

$$\texttt{readint} : \texttt{int IO} \tag{1}$$
$$\texttt{writeint} : \texttt{int} \to \texttt{unit IO} \tag{2}$$

7. Assume `MLreadint` and `MLwriteint` are primitives with side effects for input/output and consider the ML expression `add1` of type `int`:

   ```
   let val x = MLreadint() in MLwriteint(x+1); x end
   ```

   Give an equivalent expression which uses the IO monad instead of side-effects, and state its type.

   Give a function `run2diff` which can be applied to the previous answer. When so applied it should give a value in the IO monad which corresponds to ML code that runs add1 twice and returns the difference between the values read.

8. State what happens when attempting to compile and execute the following Java fragment (explaining the origin of any error messages or exceptions which might arise).

   ```
   Object n = new Integer(42), o = new String("Whoops");
   Object [] v;
   Integer [] w = new Integer[10];
   v = w;
   v[4] = n;
   v[5] = o;
   ```

9. Consider the Java code:

   ```
   Object n = new Integer(42);
   ArrayList<? extends Object> v1;
   ArrayList<Object> v2;
   ArrayList<Integer> w = new ArrayList<>(10);
   ```

   Explain any differences in behaviour between assignments `v1 = w` and `v2 = w` and also between method calls `v1.set(4,n)` and `v2.set(4,n)`.

10. In the programming language Scala, a generic class like the following one

    ```
    abstract class Stack[A] {
      def push(x : A) : Stack[A];
      def top : A;
      def pop : Stack[A];
    }
    ```

    is non-variant by default. Why? Modify it to make it co-variant.

11. Consider the declarations

    ```
    structure Z = struct   type t = int; val z = 0 end;
    structure A = Z :  sig type t;       val z : t end;
    structure B = Z :> sig type t = int; val z : t end;
    structure C = Z :> sig type t;       val z : t end;
    ```

    in the SML Modules language. Explain the behaviour of the SML interpreter on inputting each of the expressions.

    ```
    Z.z = A.z;
    Z.z = B.z;
    Z.z = C.z;
    ```

12. What are the similarities and differences between Haskell type classes, overloading, and dynamic dispatch used in object-oriented languages?