1. **What is referential transparency? Why is it important?**

Fungsi tersebut tidak memiliki satu nilai, karena nilainya tergantung dari parameter dari fungsi tersebut. Dalam pure functional language semua fungsinya itu referentially transparent (akan selalu menghasilkan nilai yang sama) selama parameternya tidak berubah.

1. **When is a function tail recursive? Why is this useful?**

Function tail recursive adalah fungsi yang memanggil rekursi dari tailnya agar tidak high cost saat menjalankan rekursinya.

1. **What is a higher-order function? An anonymous function?**

Function adalah awal dari suatu class, dimana bisa untuk membuat sebuah fungsi baru dengan menambahkannya. Fungsi ini menerima fungsi callback sebagai argumen ataupun value sebagai output.

1. **What are curried functions? Why are they useful?**

Menggunakan satu argument dalam satu waktu. Teknik mengubah **fungsi** dengan multiple parameter/argumen menjadi pecahan banyak **fungsi**, tiap **fungsi** harus mengambil setiap parameter yang ada. Jadi keuntungannya ialah tidak akan boros (pada line of code).cd

1. **How can you avoid recalculating values in a multiply recursive function?**

Dengan menggunakan recursive ditentukan oleh operasi sebelumnya, maka di kalkulasikan di operasi selanjutnya sesuai dengan logika dan keinginan (dengan memanggil fungsi tersebut kembali). Contoh pada bilangan fibonachi

1. **What is lazy evaluation?**

Lazy evaluation tidak dicek secara terus menerus. Hanya mengecek function yang di pakai, kalaupun ada yang error nanti akan dipanggil jika dipanggil fungsinya. Di cek hanya ketika dibutuhkan.

1. **What are lazy lists?**

Infinite data structure dimana value nya di generate jika ingin (sesuai kebutuhan). Tidak akan di cek satu persatu listnya, di deklarasikan dahulu saja belum tentu nanti akan di panggil.

1. **Why don’t pure functional languages provide loop constructs?**

Karena proses abstraksinya berbeda, pada pure functional languages menggunakan fungsi recursive. Akan tetapi, recursive yang di pakainya pada tail recursive.

Hasil abstraksi fungtional language melalui pendekatan fungsi matematis. Namun, dalam lingkup lain berbasis penyelesaian masalah biasa.

1. **When would you use patterns rather than guards to specify functions?**

Ketika pilihan dari pencocokan polanya lebih dari sekedar 2 pilihan. Karena guards dipilih jika pilihannya dibuat dengan ekspresi Boolean.

1. **Can you build a list that contains both numbers and functions?**

Tidak, karena element list pada haskell haruslah bertype sama.

1. **How would you simplify fibs so that (a+b) is only called once?**

Dengan menggunakan recursive (bukan tail recursive) yang ditentukan oleh operasi sebelumnya, maka di kalkulasikan di operasi selanjutnya sesuai dengan logika dari hasil angka fibonachi tersebut.

1. **What kinds of applications are well-suited to functional programming**

Functional programming cocok untuk setiap masalah yang ada, terutama untuk OOP (object oriented programming) karena kebanyakan berasal dari latar belakang matematika. Selain itu Functional Programming pun banyak digunakan dalam membuat AI (artificial intelligence) terutama Lisp.