iOS Engineer Coding Test

1. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
enum MyError: Error {
    case broken
    case busted
    case badgered
}

func willThrowAnError() {
    throw MyError.busted
}

do {
    try willThrowAnError()
} catch MyError.busted {
    print("Code was busted!")
} catch {
    print("Code just didn't work")
}
```

Answer:

Jika code tersebut di run, maka error yang akan di throw **tidak akan bisa ditangkap** oleh blok try-catch nya. Hal ini terjadi karena fungsi **willThrowAnError** bukanlah sebuah throwable function yang bisa melemparkan error untuk ditangkap di block catch nya. Agar fungsi tersebut bisa melemparkan error dari enum **MyError** yang conform terhadap protocol NSError (sekarang di rename jadi Error saja) maka fungsi **willThrowAnError** perlu diubah menjadi sebuah throwing function dengan menambah **throws** setelah parameter fungsi tersebut.

2. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
var names = [String]()
names.reserveCapacity(2)
names.append("Amy")
names.append("Rory")
names.append("Clara")
```

Answer:

Hasil dari kode tersebut adalah sebuah array yang berisi **String**, dengan 3 buah elemen “Amy”, “Rory”, dan “Clara”. Penggunaan **reserveCapacity()** diatas sebenernya boleh dilakukan ketika kita tahu persis berapa jumlah elemen dari sebuah array. Hal ini dilakukan untuk menghindari realokasi array di memory berkali kali yang bikin boros resource. Kalaupun ada penambahan dari reserve awal, array masih bisa menampung elemen baru. Tapi ketika kita tidak tahu bakal ada berapa data yang masuk ke array, mending menghindari pemanggilan **reserveCapacity()** yang justru akan menjadi masalah performance. Kalau datanya dinamis, tinggal pakai **append()** saja untuk menambahkan elemen kedalam array.

3. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
func greet(names: String...) {
    print("Criminal masterminds:", names.joined(separator: ", "))
}

greet(names: "Malcolm", "Kaylee", "Zoe")
```

Answer:

Outputnya adalah

Criminal masteminds:Malcolm, Kaylee, Zoe

Fungsi tersebut adalah sebuah **variadic function**, yang menerima inputan berupa **string** yang jumlahnya lebih dari satu. Parameter tersebut kemudian dapat diakses didalam fungsi sebagai sebuah array. Di fungsi greet tersebut, array dari input parameter akan di print dan elemen nya semua digabung dengan separator berupa koma dan spasi.

4. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
let userLoggedIn: Bool? = false

if !userLoggedIn! {
    print("Message one")
} else {
    print("Message two")
}
```

Answer:

Output nya adalah

Message one

Di line pertama, di declare userLoggedIn sebagai sebuah optional type, artinya bisa ada nilai nya dan bisa juga null. Namun, di line pertama juga langsung di assign dengan nilai false sehingga tetap bertipe optional namun telah punya nilai (false). Di kondisi **if** case, tanda seru sebelum variable merupakan **not** sedangkan tanda seru dibelakang adalah teknik untuk **force unwrapping**. Teknik ini digunakan untuk mengubah variable dengan tipe optional menjadi sebuah variable yang pasti mempunyai value. Ketika dalam kondisi tahu persis variable tersebut mempunyai value, force unwrappign sebenernya boleh digunakan, namun akan lebih safe ketika menggunakan cara optional binding lain seperti dengan menggunakan **if-let** atau **guard-let.**

5. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
func fizzbuzz(number: Int) -> String {
    switch (number % 3 == 0, number % 5 == 0) {
    case (true, false):
        return "Fizz"
    case (false, true):
        return "Buzz"
    case (true, true):
        return "FizzBuzz"
    default:
        return String(number)
    }
}

print(fizzbuzz(number: 15))
```

Answer:

Outputnya adalah

FizzBuzz

Karena 15 dibagi 3 sisanya 0, dan 15 dibagi 5 sisanya 0. Code nya rapi dengan menggunakan **tuple,** sisa bagi 3 dan sisa bagi 5 dibentuk sebagai sebuah tuple **(Bool, Bool)** yang digunakan dalam switch, sehingga untuk membagi case nya bisa langsung dengan tuple yang terdiri dari 2 boolean juga.

6. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
let names = ["Pilot": "Wash", "Doctor": "Simon"]
let doctor = names["doctor"] ?? "Bones"
print(doctor)
```

Answer:

Outputnya adalah

Bones

Hal tersebut terjadi karena di array yang terdiri dari 2 elemen **dictionary**, tidak ada value untuk elemen dengan key **doctor**. Karena mendapatkan return nil dengan memanggil

names[“doctor”] (karena case sensitive), maka didapat **default value** yang sudah di set dengan **default operator (??)**.

7. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
var spaceships = Set<String>()
spaceships.insert("Serenity")
spaceships.insert("Enterprise")
spaceships.insert("TARDIS")
spaceships.insert("Serenity")
print(spaceships.count)
```

Answer:

Outputnya adalah

3

Dalam **Set** di swift, isinya adalah entitas yang unik sehingga Menjamin tidak ada data yang sama. Contohnya ketika diawal sudah insert serenity, maka ketika insert serenity selanjutnya bakal tidak mempengaruhi isi dalam set karena serenity Sudah ada. Oiya, set dalam swift isinya bersifat unordered list. Jadi ketika butuh penyimpanan data yang tidak memperdulikan order dan harus bersifat unik (setiap data Hanya ada satu) maka Set adalah pilihan terbaik.

8. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
import Foundation
let crew = NSMutableDictionary()
crew.setValue("Kryten", forKey: "Mechanoid")
print(crew.count)
```

Answer:

Outoutnya adalah

1

Code tersebut Adalah membuat sebuah dictionary dengan tipe NSMutableDictionary (mutable, bersifat bisa diubah). Kemudian mengeset value nya dengan sebuah object

String “Kryten” dengan key “Mechanoid”. Sehingga ketika di print count nya, maka hasilnya adalah satu. Untuk mendapatkan kembali value Kryten yang berupa string, bisa dipanggil dengan cara **print(crew[“Mechanoid”])**

9. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
let numbers = [1, 3, 5, 7, 9]
let result = numbers.map { $0 * 10 }
print(numbers)
```

Answer:

Outputnya adalah

[10, 30, 50, 70, 90]

Code tersebut adalah sebuah contoh salah satu higher order function dalam sebuah collection type yaitu map di sebuah array. Map akan melakukan iterasi pada array, setiap elemennya akan dilemparkan kedalam sebuah closure (dalam hal ini adalah dikali 10), kemudian akan memberikan return array baru dari array yagn telah dilakukan iterasi. Hal ini lebih mudah dilakukan daripada harus menggunakan for atau forin.

10. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
let foo = 0x10
print(foo)
```

Answer:

16

Code diatas merupakan contoh deklarasi sebuah integer literal dengan base hexadecimal. Secara default, integer literal direpresentasikan dengan desimal (0-9). Namun ketika ingin menggunakan base lain, bisa dengan binary yang ditandai dengan **0b**, dengan octal **0o**, atau dengan hexadesimal seperti contoh diatas dengan **0x**.

11. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
struct Spaceship {
    var name: String

    func setName(_ newName: String) {
        name = newName
    }
}
var enterprise = Spaceship(name: "Enterprise")
enterprise.setName("Enterprise A")
print(enterprise.name)
```

Answer:

Error.

Dari code diatas, kita membuat sebuah struct Spaceship, yang mempunyai fungsi untuk mengubah nilai dari property struct tersebut. Dan di swift, hal ini tidak bisa dilakukan. Struct secara default melarang kita untuk mengubah propertinya secara langsung didalam fungsinya sendiri. Nah untuk bisa mengubah properti dalam fungsinya sendiri, dibutuhkan keyword **“mutating”** di depan fungsi, menjadi **mutating func setName(_ newName: String)** agar bisa jalan.

12. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
let possibleNumber = "1701"
let convertedNumber = Int(possibleNumber)
```

Answer:

1701

Line pertama, possibleNumber merupakan sebuah String dnengan value “1701”, yang akan diconvert menjadi Integer dengan fungsi **Int()** yang akan menghasilkan integer yang bersifat optional (bisa ada valuenya, atau nil / tidak bisa diconvert). Sehingga ketika ingin mengakses convertedNumber dalam sebuah integer, bisa dilakukan dengan

```
if let intNumber = convertedNumber {
    print(intNumber)
}
```

13. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
let name = "Simon"

switch name {
case "Simon":
    fallthrough

case "Malcom", "Zoe", "Kaylee":
    print("Crew")

default:
    print("Not crew")
}
```

Answer:

Crew

Hal ini dikarenakan ada keyword **fallthrough** yang berarti akan masuk kedalam case selanjutnya, walaupun sebenarnya kondisi nya tidak masuk dalam case tersebut tetapi pasti masuk dan tetap execute blok di case selanjutnya.

14. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
let numbers = [1, 3, 5, 7, 9]
let result = numbers.filter { $0 >= 5 }
```

Answer:

Result = [5, 7, 9]

Array result didapat dari fungsi filter di array numbers. Filter ini juga merupakan high order function, temannya map, reduce, dan flatmap. Filter akan menghasilkan array hasil dari iterasi, dari setiap elemen akan diambil yang sesuai dengan kondisi untuk dibentuk jadi sebuah array baru.

15. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
class Starship {
    var name: String

    override init(initialName: String) {
        name = initialName
    }
}

let serenity = Starship(initialName: "Serenity")
print(serenity.name)
```

Answer:

Error. Class Starship tidak punya parent, sehingga tidak ada initializer yang bisa di override.

16. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
let numbers = Array(1..<10)
print(numbers.count)
```

Answer:

9

Number didapat dengan initializer array yang mengambil nilai integer antara 1 hingga kurang dari 10 sehingga array tersebut isinya **[1,2,3,4,5,6,7,8,9]** dan di line kedua di print jumlah elemen dalam array tersebut.

17. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
let people = [String](repeating: "Malkovitch", count: 2)
print(people)
```

Answer:

["Malkovitch", “Malkovitch"]

Array people didapat dengan array literal yang mempunyai init parameter **repeating:count** yang akan membuat elemen sebanyak **count** dan isinya adalah value dari **repeating.**

18. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
let point = (556, 0)
switch point {
case (let x, 0):
    print("X was \(x)")
case (0, let y):
    print("Y was \(y)")
case let (x, y):
    print("X was \(x) and Y was \(y)")
}
```

Answer:

X was 556

Dengan menggunakan tuple, point di deklarasikan dan di assign dengan (556, 0). Dari tuple point tesebut, jadi beberapa case di switch. Misalnya **case (let x, 0)** artinya apapun yang pertama dari tuple, selama yang kedua 0 berarti masuk dalam kondisi tersebut. Kemudian **case (0, let y),** apapun **y** nya selama **x** nya 0 maka akan sesuai dengan kondisi.

19. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
let numbers = [1, 2, 3].map { [$0, $0] }
```

Answer:

[[1, 1], [2, 2], [3, 3]]

Array 2 dimensi teesebut didapat dari hasil map. Map akan melakukan iterasi ke semua elemen, kemudian masing masih elemen akan diolah sesuai dengan closure nya. Di kasus atas ini, setiap elemen atau $0 akan dimasukkan kedalam sebuah array dengan 2 elemen yang sama sehingga didapat array 2 dimensi seperti di output.

20. Apa output yang akan dihasilkan oleh kode di bawah ini? Jelaskan jawabanmu!

```
for i in stride(from: 1, to: 17, by: 4) {
    print(i)
}
```

Answer:

1

5

9

13

Code diatas adalah iterasi antara 1 hingga 17, namun akan loncat sebanyak 4. Dari yang pertama, masuk angka 1, kemudian selang 4, masuk 5, selang 4 lagi, dan seterusnya hingga maksimal 17.

21. Jelaskan bagaimana menangani data dari backend jika data tersebut nil?

Hal ini tergantung apakah data ini penting atau tidak. Ketika memang datanya penting, dan ada kemungkinan null, maka langkah pertama yang bisa diambil adalah dengan membuat properti dalam model sebagai sebuah **optional type**. Dengan begitu, ketika data nil tidak akan membuat aplikasi crash/force close. Selanjutnya, untuk mengakses data dengan tipe optional tesebut, kita bisa menggunakan **if-let** atau **guard-let** sehingga ketika mendapatkan nil, kita bisa execute blok code sesuai dengan yang dikehendaki.

22. Jelaskan apa yang anda ketahui tentang App Life Cycle pada iOS?

Lifecycle sebuah aplikasi dimanage di class **AppDelegate**. Di class tersebut, dihandle transisi dari setiap application state dibawah ini:

- **Not running**, ketika aplikasi belum pernah diaktifkan
- **Inactive**, ketika aplikasi berada di foreground namun tidak sedang menerima event apapun
- **Active**, ketika aplikasi berada di foreground dan sedang menerima event. State ini merupakan state ketika aplikasi biasanya sedang digunakan oleh user
- **Background**, ketika aplikasi berada di background dan sedang mengeksekusi event tertentu. Misalnya sinkronisasi data.
- **Suspended**, ketika aplikasi berada di background dan tidak mengeksekusi event apapun

Dari application state tersebut, transisinya diatur dalam class appDelegate dalam beberapa method:

- **application: didFinishLaunchingWithOptions**

Method ini adalah yang pertama kali dipanggil ketika aplikasi pertama launch, dari state not running ke active.

- **applicationWillEnterForeground:**

Method ini dipanggil setelah **application: didFinishLaunchingWithOptions:**, atau akan dipanggil ketika aplikasi kembali ke state active. Misalnya ketika aplikasi terinterupt oleh panggilan telepon, atau system interrupt lain.

- **applicationDidBecomeActive:**

Method ini dipanggil setelah **applicationWillEnterForeground:** selesai, biasanya ketika transisi ke foreground selesai dan pindah ke state active.

- **applicationWillResignActive:**

Method ini dipanggil ketika akan pindah ke state inactive, misalnya ketika user memencet home button

- **applicationDidEnterBackground:**

Method ini akan dipanggil ketika dari status inactive menjadi ke background.

- **applicationWillTerminate:**

Method ini akan dipanggil ketika aplikasi berada di status suspended dan akan di purge dari memory untuk memberikan space ke aplikasi yang lain agar performa iphone bagus

23. Jelaskan bagaimana mengatasi ukuran tampilan apps yang berbeda device?

Dengan perkembangan saat ini yang begitu cepat sekaligus banyak pilihan ukuran device, tentunya ukuran tampilan bakal menjadi challenge tersendiri untuk developer. Beberapa teknik bisa digunakan, intinya adalah membuat layout yang bisa adaptive terhadap ukuran layar.

- Dengan menggunakan auto layout untuk mengatur tampilan. Auto layout menggunakan constraint untuk menata object UI di sebuah layar. Constraint tersebut akan menentukan bagaimana object beradaptasi ketika ukuran layar berubah.
- Dengan menggunakan Size Classes. Size classes ini akan membagi tampilan menjadi beberapa kelompok, dan kita harus memberikan ukuran berbeda untuk setiap kelompok tersebut.

24. Sebuah restoran cepat saji setiap harinya melayani customer yang berbeda-berbeda. Manager dari restoran tersebut ingin mengetahui siapa saja customer yang datang ke restorannya dengan rincian