Cho biết kết quả của biểu thức sau viết bằng Scala:

List(3,5,9,10).foldLeft(0)((x,y)=>x+y)

- (●) 27
- ( ) List(27)
- ( ) List(3,5,9)
- ( ) 3

Cho filter là hàm bậc cao nhận vào một vị từ (predicate - hàm có kết quả kiểu luận lý) và trả về một danh sách con gồm các phần tử thoả vị từ đó và cho phép toán % là phép toán tính modulo. Hãy viết biểu thức để trả về một danh sách con chỉ gồm các giá trị chẵn của danh sách vào? List(2,3,4,5,6).__________ => List(2,4,6)

- ( ) filter(x => x % 2)
- ( ) filter(x % 2 == 0)
- (●) filter(x => x % 2 == 0)
- ( ) filter((x,y)=> x % 2 == y)

Cho một danh sách List(4,2,6,9), hãy cho biết dùng hàm bậc cao nào để tạo được một danh sách là các giá trị bình phương của các phần tử trong danh sách được cho, tức List(16,4,36,81)

- ( ) foldLeft
- (●) map
- ( ) filter
- ( ) forall

Cho reverse là hàm đảo ngược thứ tự các phần tử của một danh sách (ví dụ List(1,4,2).reverse => List(2,4,1)) và :: dùng để nối 1 phần tử vào đầu một danh sách (a :: List(b,c,d) => List(a,b,c,d)). Với lst đang chứa một danh sách các giá trị nguyên, cho biết biểu thức nào dưới đây có kết quả tương tự reverse?

- ( ) lst.map(x => x :: List())
- ( ) lst.foldLeft(List[Int]())((x,y)=>x::y)
- ( ) lst.forAll(x => x::List())
- (●) lst.foldLeft(List[Int]())((x,y) => y::x)

Hãy viết một hàm (forAllExist) nhận vào 3 thông số gồm 1 danh sách các số nguyên và 2 predicate, chỉ trả về kết quả true nếu danh sách có tất cả phần tử thoả predicate thứ nhứt và có ít nhất 1 phần tử thoả predicate thứ hai.

- (●) def forAllExist(lst:List[Int],f1:Int=>Boolean,f2:Int=>Boolean) = lst.forall(f1) && lst.exists(f2)
- ( ) def forAllExist(lst:List[Int],f1:Boolean,f2:Boolean) = lst.forall(f1) && lst.exists(f2)
- ( ) def forAllExist(lst:List[Int],f1:Int,f2:Int) = lst.forall(f1) && lst.exists(f2)
- ( ) def forAllExist(lst:List[Int],f1:Boolean=>Boolean,f2:Boolean=>Boolean) = lst.forall(f1) && lst.exists(f2)

Hãy viết một hàm (doubleCheck) nhận vào 3 thông số gồm 1 danh sách các số nguyên và 2 predicate, chỉ trả về kết quả true nếu danh sách có ít nhất một phần tử thoả predicate thứ nhứt và có ít nhất một phần tử thoả predicate thứ hai.

- ( ) def doubleCheck(lst:List[Int],f1:Int=>Boolean,f2:Int=>Boolean) = lst.exists(x => f1(x) && f2(x))
- (●) def doubleCheck(lst:List[Int],f1:Int=>Boolean,f2:Int=>Boolean) = lst.exists(f1) && lst.exists(f2))
- ( ) def doubleCheck(lst:List[Int],f1:Int=>Boolean,f2:Int=>Boolean) = lst.forall(x => f1(x) && f2(x))
- ( ) def doubleCheck(lst:List[Int],f1:Int=>Boolean,f2:Int=>Boolean) = lst.forall(f1) && lst.forall(f2))

Cho một hàm được định nghĩa như sau:

def increaseClosures(n:Int)(x:Float) = x + n

Một lênh khai báo biến inc3 được viết như sau:

val inc3 = increaseClosures(3) _

sẽ làm cho inc3 được cất giữ giá trị gì?

- ( ) Không cất giữ gì cả vì lệnh gọi hàm increaseClosures bị sai, không đủ thông số
- ( ) 3
- ( ) x + 3
- (●) Một hàm có kiểu là Float => Float

Cho định nghĩa hàm như sau:

def foo(x:Int)(y:Float) = x * y

Khi định nghĩa x như sau: val x = foo(3) _ thì x sẽ có kiểu là gì

- ○ Int
- ○ Float
- ○ Không có kiểu gì cả vì việc gọi hàm foo bị sai do thiếu thông số
- ◉ Kiểu hàm: Float => Float

Cho hàm foo được định nghĩa như sau:

def foo(x:Boolean,y:Float)(z:Float)= if (x) y else z

và m được dịnh nghĩa như sau: val m = foo(true) _ _

Cho biết kiểu của m?

- ○ Float
- ◉ Không có kiểu gì cả vì lệnh gọi foo chưa đúng
- ○ Kiểu hàm Float => Float
- ○ Kiểu hàm Float => Float => Float

Cho listA là một danh sách có 3 phần tử {a, b, c} và listB là một danh sách có 3 phần tử là {d,e,f}, nếu tác vụ listA.append(listB) sẽ làm cho listA trở nên có 6 phần tử gồm 3 phần tử ban đầu của listA và 3 phần tử của listB thì tác vụ append là?

- ○ Immutable
- ◉ Mutable

Chính xác. tác vụ append làm thay đổi listA nên nó là một tác vụ mutable.

Cho x chứa chuỗi "abc", nếu x.toUpper sẽ làm cho x trở thành chứa chuỗi "ABC" thì toUpper là?

- ○ Immutable
- ◉ Mutable

Cho listA là một danh sách có 3 phần tử {a, b, c} và listB là một danh sách có 3 phần tử là {d,e,f}, nếu tác vụ listA.append(listB) trả về một danh sách mới có 6 phần tử gồm 3 phần tử của listA và 3 phần tử của listB, trong khi listA và listB không đổi, thì tác vụ append là?

- (●) Immutable
- ( ) Mutable

Chính xác, tác vụ append không làm thay đổi các thông số của nó nên append là immutable.

Giả sử Scala là một ngôn ngữ lập trình hàm tinh khiết (pure functional programming language) và giả sử Scala có phát biểu while như sau:

while (a < 1) { ... }

Thân vòng lặp là một hộp đen, không rõ về nội dung. Bạn có thể suy luận gì về lệnh lặp trên?

- ( ) Không thực thi được lần nào
- ( ) Sẽ thực thi bình thường như một lệnh lặp while trên các ngôn ngữ khác (Java)
- ( ) Luôn luôn lặp vô hạn
- (●) Hoặc không thực thi hoặc lặp vô hạn

Đúng, trên ngôn ngữ lập trình hàm thuần khiết, các biến đại diện cho một giá trị không đổi, do đó, biểu thức a < 1 hoặc là luôn luôn sai hoặc là luôn luôn đúng nên lệnh while sẽ hoặc là không được thực thi (khi a < 1 sai) hoặc là lặp vô hạn (khi a < 1 đúng). Vì vậy trên các ngôn ngữ lập trình hàm thuần khiết, không có lệnh lặp dựa vào biểu thức điều kiện như while, do while.

Trên Scala, phép toán :: dùng để nối 1 phần tử vào 1 danh sách (3::List(2,1,5) sẽ tạo thành danh sách List(3,2,1,5)). Giả sử biến sl đang chứa giá trị List(3,5,6,2), hãy cho biết phép match sau có thành công không và nếu có thì giá trị của h và t là bao nhiêu?

```
sl match {
    case h :: tl => ....
```

- (●) Thành công, h là 3 và t là List(5,6,2)
- ( ) Không thành công, phải ghi case List(3,5,6,2) thì mới thành công
- ( ) Không thành công, phải ghi case List(t,h) mới thành công
- ( ) Nếu h đang là 3 và t đang là List(5,6,2) thì mới thành công

Trên Scala, kiểu tuple là kiểu kết hợp nhiều giá trị với nhau, ví dụ val m = (3,4.3, True) là một giá trị kiểu tuple kết hợp 3 kiểu Int, Double và Boolean. Cho biết với m định nghĩa trong ví dụ trên, trong phép match sau thì thành công ở case nào?

```
m match {
  case (3,1.0,True) => ...
  case (3,4.3,False) => ...
  case (_,_,_) =>
}
```

- ( ) Không thành công ở case nào cả
- ( ) Thành công ở case đầu tiên vì có giá trị 3 trùng nhau
- ( ) Thành công ở case thứ hai
- (●) Thành công ở case cuối

Cho một khai báo hàm như sau:

def foo(x:Boolean,y:Int,z:Int) = if (x) y else z

Hỏi khai báo trên sẽ gây ra lỗi gì?

- ○ Không gây ra lỗi gì
- ○ Lỗi biên dịch vì không khai báo kiểu trả về
- ◉ Có thể gây ra lỗi khi thực thi tuỳ theo cách gọi hàm
- ○ Luôn gây ra lỗi runtime

Đúng, tuỳ theo cách gọi hàm, có thể xảy ra lỗi khi thực thi. Ví dụ foo(a==0,1,b/a) sẽ gây ra lỗi chia cho 0 khi a bằng 0. Hoặc foo(x == null,null,x.m()) sẽ gây ra lỗi Null pointer reference.

Để khắc phục lỗi về ngữ dụng trong bài tập 1, cần phải sửa khai báo hàm như thế nào?

- ○ def foo(x:Boolean)(y:Int)(z:Int) = if (x) y else z
- ○ def foo(x:Boolean,y:Int) = (z:Int) => if (x) y else z
- ◉ def foo(x:Boolean, y: => Int, z: => Int) = if (x) y else z
- ○ def foo(x:Boolean,y: Int, z:Int) = {

```
    lazy val m = y
    lazy val n = z
    if (x) m else n
}
```

Đúng, khi viết => trước kiểu của thông số, thì thông số sẽ được truyền theo tên (pass-by-name). Khi đó các biểu thức thông số chỉ được tính khi thông số hình thức được sử dụng trong thân hàm.

Cho biết kiểu trả về của hàm sau:

def foo(a:Boolean) = if (a) 1 else List( )

- ○ Int
- ○ List
- ○ AnyVal
- ◉ Any

Đúng, trên cây phân lớp của Scala, Any là lớp gốc nên nó là lớp tổ tiên của cả lớp Int và List

Chọn khai báo hàm đúng và ngắn nhất trong các khai báo sau:

- ◉ def foo(x:Int):Int = if (x == 0) 1 else x*foo(x-1)
- ○ def foo(x):Int = x + 3
- ○ def foo(x:Int):Int = x + 4
- ○ List(1,2,3).map((x:Int)=> x + 1)

Đúng, hàm đệ qui nên cần phải khai báo kiểu trả về

Chọn biểu thức đúng và ngắn nhất trong các biểu thức sau:

- ○ List(4,5,6).forall((x:Int) => x > 3)
- ○ List(4,5,6).forall(x => x > 3)
- ◉ List(4,5,6).forall(_ > 3)
- ○ List(4,5,6).forall(> 3)

Đúng