Recitation Session October 04 2016

Please do not write on this sheet of paper
And do not use laptops during the session

Function values

This week we will work on playing with functions as values.

Ex 1.

Define the function `flip`. It takes a function and returns the same function, but with the arguments flipped.

```
def flip(f: (Int, Double) => Int): (Double, Int) => Int = ???
```

Ex 2.1

Define the identity function for integers, which, given an `Int`, returns it

```
val id: Int => Int = ???
```

Ex 2.2

Define the `compose` function, that, given 2 functions f, g, returns a function that composes them, i.e., f ∘ g.

```
def compose(f: Int => Int, g: Int => Int): Int => Int = ???
```

What does `compose(id, f)(k)` evaluate to, for some function `f` and integer `k` ?

Ex 2.3

Define the function `repeated`, which takes a function and repeatedly applies it `n` times (`n ≥ 0`).

```
def repeated(f: Int => Int, n: Int): Int => Int = ???
```

Hint: What values should be returned by `repeated(x => x + 1, 0)` and `repeated(x => x + 1, 3)` ?

Ex 3.

Write a function `fixedPoint` with the following signature:

```
def fixedPoint(f: Int => Int): Int => Int
```

The function takes a function `f` and returns a function that applies `f` up until it reaches a fixed point. A value `x` is a fixed point of `f` if `f(x) == x`.

For each of the following expressions, indicate whether it terminates, and if so, what is the value returned:

- `fixedPoint(id)(123456)`
- `fixedPoint(x => x + 1)(0)`
- `fixedPoint(x => if (x % 10 == 0) x else x + 1)(35)`
- `fixedPoint((x: Int) => x / 2 + 5)(20)`

Ex 4.1

Define the function `curry2`, that curries a two arguments function.

```
def curry2(f: (Int, Int) => Int): Int => (Int => Int) = ???
```

Hint: what should `curry2((x, y) => x + y)(1)` return?

Ex 4.2

Define the function `uncurry2`. It takes a curried function, and creates a two-argument function.

```
def uncurry2(f: Int => Int => Int): (Int, Int) => Int = ???
```

Ex 5.1

Write the `sum` function with the following signature:

```
def sum(a: Int, b: Int)(f: Int => Int): Int = ???
```

Which returns the following value: $\sum_{i=a}^{b-1} f(i)$

Bonus point: Can your implementation be tail recursive ?

Ex 5.2

Write the `quadratic` function with the following signature:

```
def quadratic(c: Int): Int => Int = ???
```

Which returns a function that takes an integer x as argument and returns $(x - c)^2$.

Ex 5.3

Using the above functions, define the function `quad3Integrate` which, given two integers a and b, outputs the following value: $\sum_{i=a}^{b-1} (i - 3)^2$

```
def quad3Integrate(a: Int, b: Int): Int = ???
val quad3Integrate: (Int, Int) => Int = ???
```