CPS Activity

Mattox Beckman

Convert To

Convert the following functions to CPS:

1.
```haskell
sumList [] = 0
sumList (x:xs) = x + sumList xs
```

2. Assume f is written in direct style.

```haskell
map f [] = []
map f (x:xs) = f x : map f xs
```

3. Assume `f` is written in CPS and takes one continuation.

```haskell
map f [] = []
map f (x:xs) = f x : map f xs
```

4. Let `f` take two continuations, one to proceed and one to abort. So, `f` takes four arguments total.

```haskell
foldr f z [] = z
foldr f z (x:xs) = f x (foldr f z xs)

-- example continuation for foldrk
timesk 0 _ k ka = ka 0
timesk _ 0 k ka = ka 0
timesk a b k ka = k (a * b)
```

Reordering Computations

5. Suppose you have a calculator which has an accumulator and a list of instructions. `Add i` adds `i` to the accumulator, and `Sub i` subtracts `i` from the accumulator.

```haskell
data Calc = Add Integer
          | Sub Integer
    deriving (Eq,Show)
```

The only problem is that our accumulator cannot be negative! Use continuations to fix this.

Here's the original calculator:

```haskell
calc xx = aux 0 xx
  where aux a [] = a
        aux a ((Add i):xs) = aux (a+i) xs
        aux a ((Sub i):xs) = aux (a-i) xs
```

Hint: you will need *two* continuations to make this work.