# 模式匹配

**1、利用模式匹配，编写一个swap函数，接受一个整数的对偶，返回对偶的两个组成部件互换位置的新对偶**

1. **利用模式匹配，编写一个swap函数，交换数组中的前两个元素的位置，前提条件是数组长度至少为2**
2. **编写一个函数，计算List[Option[Int]]中所有非None值之和。不得使用match语句。**

# 高阶函数

**1、编写一个compose函数，将两个类型为Double=>Option[Double]的函数组合在一起，产生另一个同样类型的函数。如果其中一个函数返回None，则组合函数也应返回None。例如：**  
**def f(x : Double) = if ( x >= 0) Some(sqrt(x)) else None  
def g(x : Double) = if ( x != 1) Some( 1 / ( x - 1)) else None  
val h = compose(f,g)  
h(2)将得到Some(1)，而h(1)和h(0)将得到None**

**2、编写函数values(fun:(Int)=>Int,low:Int,high:Int),该函数输出一个集合，对应给定区间内给定函数的输入和输出。比如，values(x=>x\*x,-5,5)应该产出一个对偶的集合(-5,25),(-4,16),(-3,9),…,(5,25)**

**3、如何用reduceLeft得到数组Array(1,333,4,6,4,4,9,32,6,9,0,2)中的最大元素?**

1. **用to和reduceLeft实现阶乘函数,不得使用循环或递归**
2. **编写函数largest(fun:(Int)=>Int,inputs:Seq[Int]),输出在给定输入序列中给定函数的最大值。举例来说，largest(x=>10x-xx,1 to 10)应该返回25.不得使用循环或递归**

**6、要得到一个序列的对偶很容易，比如:**

**val pairs = (1 to 10) zip (11 to 20)**

**编写函数adjustToPair,该函数接受一个类型为(Int,Int)=>Int的函数作为参数，并返回一个等效的, 可以以对偶作为参数的函数。举例来说就是:adjustToPair(\_\*\_)((6,7))应得到42。然后用这个函数通过map计算出各个对偶的元素之和**

**7、实现一个unless控制抽象，工作机制类似if,但条件是反过来的**