做完第一个大作业，发现标准答案的思路和我的不太一样，我的思维还是 imperative programming 的思维，要慢慢适应 functional programming 的思维。

1) 我的答案，即便是在只用一次的情况下，我都是新建一个变量保存函数返回值，然后再对变量进行操作。而标准答案是直接利用函数返回值了。不得不说，没有中间变量反而更好读一些。如下图所示，上面是我的，下面是标准答案的。

```
fun dates_in_months(dates : (int * int * int) list, months : int list)=
    if null dates orelse null months
    then []
    else
    let
        val prelist = dates_in_months(dates, tl months)
        val curlist = dates_in_month(dates, hd months)
    in
        curlist @ prelist
    end
```

```
1  fun dates_in_months(dates : (int * int * int) list, months : int list) =
2      if null months
3      then []
4      else dates_in_month(dates, hd months) @ dates_in_months(dates, tl months)
5
```

2) 要灵活使用函数退出条件。比如如下图所示的代码，退出条件是 day1>day2，这时候返回是一个空列表。其实我在后面递归的时候可以利用这个退出条件来简化代码。当迭代到最后一步的时候 day1=day2，此时继续迭代下去返回的是空列表，这样我的代码 else if 和 else 两个分支可以合并在一起。

```
fun month_range(day1 : int, day2 : int)=
    if day1 > day2
    then []
    else if day1 = day2
    then [what_month(day2)]
    else
    let
        val cur_month = what_month(day1)
        val pre_months = month_range(day1 + 1, day2)
    in
        cur_month :: pre_months
    end
```

```
1  fun month_range (day1 : int, day2 : int) =
2      if day1 > day2
3      then []
4      else what_month day1 :: month_range(day1 + 1, day2)
```

3） 我用了很多冗余 if then else，而答案直接是直接把用到的变量值计算好，然后直接丢逻辑，这样确实更好读一些。而且代码量会少很多，下面是两个算法的对比。而且对于返回值是 bool 的函数，直接将表达式作为返回就可以了，免得 if 一下，不是多此一举画蛇添足吗。

```
fun is_older(right : int * int * int, left : int * int * int)=
    if (#1 right < #1 left) orelse ((#1 right = #1 left) andalso (#2 right < #2 left)) orelse ((#1 right = #1 left) andalso (#2 right = #2 left) andalso (#3 right < #3 left))
    then true
    else false
```

```
1  fun is_older (date1 : int * int * int, date2 : int * int * int) =
2      let
3          val y1 = #1 date1
4          val m1 = #2 date1
5          val d1 = #3 date1
6          val y2 = #1 date2
7          val m2 = #2 date2
8          val d2 = #3 date2
9      in
10         y1 < y2 orelse (y1=y2 andalso m1 < m2)
11                 orelse (y1=y2 andalso m1=m2 andalso d1 < d2)
12     end
```

```
fun reasonable_date(date : int * int * int)=
    let
    fun isleap(year : int)=
        if (year mod 400 = 0) orelse (year mod 4 = 0 andalso year mod 100 <>0)
        then true
        else false

    val months = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    (* get the length of a month *)
    fun getmonth(month : int, months : int list)=
        if month = 1
        then hd months
        else getmonth(month-1, tl months)

    in
    if ((#1 date) <= 0) orelse ((#2 date) < 1) orelse ((#2 date) > 12) orelse ((#3 date) < 1)
    then false
    else
        let
        val month_length = getmonth(#2 date, months)
        in
        if isleap(#1 date) andalso ((#2 date) = 2)
        then
            if (#3 date) > month_length + 1
            then false
            else true
        else
            if (#3 date) > month_length
            then false
            else true
        end
    end
```

```
22 fun reasonable_date (date : int * int * int) =
23     let
24         fun get_nth (lst : int list, n : int) =
25         if n=1
26         then hd lst
27         else get_nth(tl lst, n-1)
28         val year  = #1 date
29         val month = #2 date
30         val day   = #3 date
31         val leap  = year mod 400 = 0 orelse (year mod 4 = 0 andalso year mod 100
           <> 0)
32         val feb_len = if leap then 29 else 28
33         val lengths = [31,feb_len,31,30,31,30,31,31,30,31,30,31]
34     in
35         year > 0 andalso month >= 1 andalso month <= 12
36         andalso day >= 1 andalso day <= get_nth(lengths,month)
37     end
```

4） 学会在用 if then else 来简化语句。比如下面两个例子：
使用一行 if then else 来判断进行求解的值，这样的好处在于不用写一大片代码，而且可读性更高些。

```
val feb_len = if leap then 29 else 28
val lengths = [31, feb_len, 31, 30, 31, 30, 31
```

```
fun number_in_month (dates : (int * int * int) list, month : int) =
    if null dates
    then 0
    else (if #2 (hd dates) = month then 1 else 0)
          + number_in_month(tl dates, month)
```

最终总结：

1） 只调用一次的函数直接使用，不必要申请中间变量。
2） 灵活使用函数退出条件。
3） 对于返回值为 bool 的函数，计算好各种取值，然后写上你想要的逻辑就行，不必要各种 if else 绕来绕去反而不好读懂。
4） 灵活使用一行 if then else 来提高代码可读性。

20181208
六 21：53
中大北实验楼
天气开始变冷