CSE341, Spring 2016, Assignment 1
Due: March 16 Wed, 11:59 pm

Submission

Write the functions in problem 1 – 5 in a single file, named "*sol1.sml*".
Then upload the file to the uni06.unist.ac.kr server (under your account), run the following command in the same directory where you have sol1.sml file.

```
plsubmit assign1 sol1.sml
```

You can submit the file multiple times, and the last one before the deadline will be used for grading. Please test submitting your solution a couple of days before the deadline to make sure the submission works for you. If it does not work, please contact the TA.

Problems

1. Merge Lists (10pt)

Write *merge* function that takes two sorted lists (in ascending order) and returns a sorted list that contains all the elements in the two lists. The signature of the function is as following:

```
merge: int list * int list -> int list
```

For example, `merge([1,4,5], [2,6,7])` should return `[1,2,4,5,6,7]`. You may assume that each of the input lists does not have repeating elements. For example, `[1,1,4,5]` cannot be used as an input to `merge` function because `1` is repeated.

2. Reverse List (10pt)

Write *reverse* function that takes a list and returns the reversed list. For example, `reverse([1,5,4])` returns `[4,5,1]`. The signature of the function is as following:

```
reverse: int list -> int list
```

For problem 2, you should not use the ML's built-in function `rev`.

3. Sigma Function (10pt)

Write *sigma* function that takes two integers, `a` and `b`, and a function `f` and returns the following:

$$\sum_{n=a}^{b} f(n)$$

The signature of *sigma* is as following:

```
sigma: int * int * (int -> int) -> int
```

In other words, `sigma(a,b,f)` computes $\sum_{n=a}^{b} f(n)$

4. Digits Function (10pt)
   Write *digits* function that takes a positive integer and returns the list of digits of the integer. For example, `digits(253)` returns `[2,5,3]`. The function's signature should be:

```
    digits: int -> int list
```

You may assume that the input is a positive number.

5. Digital Roots and Additive Persistence (20pt: 10pt for each function)
   Consider the process of taking a number, adding its digits, then adding the digits of the number derived from it, etc., until the remaining number has only one digit. The number of additions required to obtain a single digit from a number `n` is called the *additive persistence* of `n`, and the digit obtained is called the *digital root* of `n`. For example, the sequence obtained from the starting number 9876 is (9876, 30, 3), so 9876 has an additive persistence of 2 and a digital root of 3. For the starting number 12349, the process produces (12349, 19, 10, 1), so 12349 has an additive persistence of 3 and a digital root of 1.
   Write two functions *additivePersistence* and *digitalRoot* that take positive integer argument `n` and return the additive persistence and the digital root of `n` respectively. The signatures of the functions are as following:

```
    additivePersistence: int -> int
    digitalRoot: int -> int
```

You can use digits function defined in the previous question.