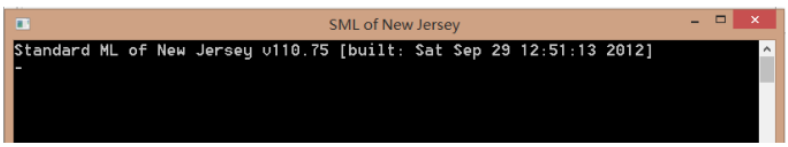
泛函程序设计原理

实验一

实验目的

- 熟悉 SML/NJ 开发环境及使用
- 掌握 SML 基本语法和书写规则
- SML 简单程序设计和程序编写

实验提示



- 在’ -‘ 提示符下直接输入 SML 语句 , 以分号结束 ;
- 表达式计算的结果缺省赋值给变量” it”;
- 文件的加载 : use <filename>;
  如 use “d:\\sml\\test.sml”;
- 程序正确性检查： val <return value> = <function> <argument value>
  如： val 42 = eval [2,4]

实验内容：

1. 在提示符下依次输入下列语句，观察并分析每次语句的执行结果。

- 3+ 4;
- 3 + 2.0;
- it + 6;
- val it = “hello”;
- it + “ world”;
- it + 5;
- val a = 5;
- a = 6;
- a + 8;
- val twice = (fn x => 2 * x);
- twice a;
- let x = 1 in x end;
- foo;
- [1, “foo”];

2. 函数 sum 用于求解整数列表中所有整数的和，函数定义如下：

```
(* sum : int list -> int   *)
(* REQUIRES: true       *)
(* ENSURES: sum(L) evaluates to the sum of the integers in L. *)
fun sum [ ] = 0;
   | sum (x ::L) = x + (sum L);
```

完成函数 mult 的编写，实现求解整数列表中所有整数的乘积。

```
(* mult : int list -> int   *)
(* REQUIRES: true       *)
(* ENSURES: mult(L) evaluates to the product of the integers in L. *)
fun mult [ ] =   (* FILL IN *)
   | mult (x ::L) =        (* FILL IN *)
```

3. 完成如下函数 Mult: int list list -> int 的编写，该函数调用 mult 实现 int list list 中所有整数乘积的求解。

```
(* mult : int list list -> int  *)
(* REQUIRES: true            *)
(* ENSURES: mult(R) evaluates to the product of all the integers in the
lists of R. *)

fun Mult [ ] =         (* FILL IN *)
   | Mult (r :: R) =   (* FILL IN *)
```

4. 函数 mult' 定义如下，试补充其函数说明，指出该函数的功能。

```
(* mult' : int list * int -> int              *)
(* REQUIRES: true       *)
(* ENSURES: mult'(L, a) ... (* FILL IN *)      *)

 fun mult' ([ ], a) = a
  | mult' (x :: L, a) = mult' (L, x * a);
```

利用 mult' 定义函数 Mult' : int list list * int -> int，使对任意整数列表的列表 R 和整数 a，该函数用于计算 a 与列表 R 中所有整数的乘积。该函数框架如下所示，试完成代码的编写。

```
fun Mult' ( [ ], a) =      (* FILL IN *)
   | Mult' (r::R, a) =     (* FILL IN *)
```

5. 编写递归函数 square 实现整数平方的计算，即 square n = n * n 。

要求：程序中可调用函数 double，但不能使用整数乘法（ * ）运算。

```
(* double : int -> int *)
(* REQUIRES: n >= 0 *)
(* ENSURES: double n evaluates to 2 * n.*)

fun double (0 : int) : int = 0
   | double n = 2 + double (n - 1)
```

6. 定义函数 divisibleByThree: int -> bool，以使当 n 为 3 的倍数时，divisibleByThree n 为 true，否则为 false 。注意：程序中不能使用取余函数' mod' 。

```
(* divisibleByThree : int -> bool  *)
(* REQUIRES: true                          *)
(* ENSURES: divisibleByThree n evaluates to true if n is a multiple of 3 and
to false otherwise *)
```

7. 函数 evenP 为偶数判断函数，即当且仅当该数为偶数时返回 true 。其代码描述如下：

```
(* evenP : int -> bool              *)
(* REQUIRES: n >= 0                 *)
(* ENSURES: evenP n evaluates to true iff n is even. *)
fun evenP (0 : int) : bool = true
  | evenP 1 = false
  | evenP n = evenP (n - 2)
```

试编写奇数判断函数 oddP: int -> bool ，当且仅当该数为奇数时返回 true 。注意：代码不要调用函数 evenP 或 mod 。