CS312 SML / NJ Cheat Sheet

```
() : unit
3 : int
3.0 : real
#"a" : char
"xyz" : string
false : bool
3 < 5 andalso true : bool
SOME 3 : int option
NONE : 'a option
ref 3 : int ref
[3,4] : int list
[] : 'a list
(1,"xyz",3.0) : int * string * real
{foo=6,bar="xyz"} : {foo:int,bar:string}
fn x => x + 1 : int -> int
fn x y => x + y : int -> int -> int
fn (x,y) => x + y : int * int -> int
fn () => 4 : unit -> int

if x < 0 orelse x > 0
  then "nonzero"
  else "zero"

case x of
  0 => "zero"
| 1 => "one"
| _ => "more than one"

Char.ord #"a"   97
Char.ord #"A"   65
Char.ord #"0"   48
Char.chr 97   #"a"
explode "abc"   [#"a",#"b",#"c"]
implode [#"a",#"b",#"c"]   "abc"

(fn x => x + 1) 3   4
"x" ^ "y" ^ "z"   "xyz"
~5 + 7  2

let
  fun f x = x * x
  val ff = f o f
  val fff = f o f o f
in
  (f 2,ff 2,fff 2)
end  (4,16,256)

hd [3,4]   3
tl [3,4]   [4]
tl [4]   [] = nil
3::[4,5]   [3,4,5]
[1,2,3] @ [4,5,6]   [1,2,3,4,5,6]
null []   true
null [3,4]   false

#2 (1,"abc",3.4)   "abc"
#foo {foo=6,bar="xyz"}   6

datatype 'a option = SOME of 'a | NONE
datatype order = LESS | EQUAL | GREATER
datatype 'a stack = EMPTY | TOP of ('a * 'a stack)

map : ('a -> 'b) -> 'a list -> 'b list
map (fn x => x + 100) [2,3,4]   [102,103,104]
map (fn x => x = 3) [2,3,4]   [false,true,false]

List.tabulate : int * (int -> 'a) -> 'a list
List.tabulate (4,fn x => x*x)   [0,1,4,9]
```

```
List.filter : ('a -> bool) -> 'a list -> 'a list
List.filter (fn x => x < 4) [2,4,3,9,6,1,0,5]   [2,3,1,0]

List.foldr : ('a * 'b -> 'b) -> 'b -> 'a list -> 'b
List.foldr (op ^) "x" ["a","b","c"]   "abcx"

List.foldl : ('a * 'b -> 'b) -> 'b -> 'a list -> 'b
List.foldl (op ^) "x" ["a","b","c"]   "cbax"

List.find : ('a -> bool) -> 'a list -> 'a option
List.find (fn x => x > 10) [1,5,10,13,19]   SOME 13

size "hello"   5
length [8,9,10]   3
rev [8,9,10]   [10,9,8]
valOf (SOME 312)   312
isSome (SOME 312)   true

let val (x,y) = (SOME 12,300)
in case (x,y) of
      (SOME z,_) => z + y
    | (NONE,_) => y
end  312

let
  val (e,pi) = (Math.e,Math.pi)
  fun f x y = {e=x,pi=y}
in f e pi
end  {e=2.71828182846,pi=3.14159265359}

let
  fun uncurried(x,y) = x + y
  fun curried x y = x + y
in (uncurried(1,2), curried 1 2)
end  (3,3)

fun sum(x:int list):int =
  case x of [] => 0
          | u::t => u + sum t

signature STACK = sig
  type 'a stack
  exception Empty of string
  val new : unit -> 'a stack
  val push : 'a stack * 'a -> 'a stack
  val pop : 'a stack -> 'a * 'a stack
  val isEmpty : 'a stack -> bool
end

structure Stack : STACK = struct
  type 'a stack = 'a list
  exception Empty of string
  fun new() = nil
  fun push(s,x) = x::s
  fun pop s =
    case s of x::t => (x,t)
            | [] => raise Empty "empty"
  val isEmpty = List.null
end

let val xr:int ref = ref 299
in xr := !xr + 13
end  ()      sets xr to 312 as a side effect

(print "hello"; 312)  312
prints "hello" as a side effect
```