```
(*
                              CS51 Lab 12
                  Imperative Programming and References
 *)

(*
Objective:

This lab provides practice with reference types and their use in
building mutable data structures and in imperative programming more
generally. It also gives further practice in using modules to abstract
data types.

There are 4 total parts to this lab. Please refer to the following
files to complete all exercises:

   lab12_part1.ml -- Part 1: Fun with references
-> lab12_part2.ml -- Part 2: Gensym (this file)
   lab12_part3.ml -- Part 3: Appending mutable lists
   lab12_part4.ml -- Part 4: Adding serialization to imperative queues
 *)

(*======================================================================
Part 2: Gensym

The `gensym` function (short for "GENerate SYMbol") has a long history
dating back to the early days of the programming language LISP.  You
can find it as early as the 1974 MacLISP manual (page 53).
http://www.softwarepreservation.org/projects/LISP/MIT/Moon-MACLISP_Reference_Manual-Apr
_08_1974.pdf

(What is LISP you ask? LISP is an untyped functional programming
language invented by John McCarthy in 1958, which he based directly on
Alonzo Church's lambda calculus. It is one of the most influential
programming languages ever devised. You could do worse than spend some
time learning the Scheme dialect of LISP, which, by the way, will be
made much easier by having learned a typed functional language --
OCaml.)

The `gensym` function takes a string and generates a new string by
suffixing a unique number, which is initially 0 but is incremented
each time `gensym` is called.

For example,

    # gensym "x" ;;
    - : string = "x0"
    # gensym "x" ;;
    - : string = "x1"
    # gensym "something" ;;
    - : string = "something2"
    # gensym (gensym "x") ;;
    - : string = "x34"

There are many uses of `gensym`, one of which is to generate new
unique variable names in an interpreter for a programming language. In
fact, you'll need `gensym` for just this purpose in completing the
final project for CS51, so this is definitely not wasted effort.

......................................................................
Exercise 4: Complete the implementation of `gensym`. As usual, you
shouldn't feel beholden to how the definition is introduced in the
skeleton code below. (We'll stop mentioning this from now on.)
......................................................................*)
```

```
let gensym (s : string) : string =
  failwith "gensym not implemented" ;;
```