```
(* Author: Flavio Lerda      *)
(* 15-212 Section C          *)
(* Problems involving stream *)
(* March, 24th 2004          *)

exception Unimplemented;

(* Problem 1:

The SML library includes a structure called TextIO which provides
basic functions for input/output. In specific we are going to look at
the function:

val TextIO.inputLine: instream -> string

NOTE: The type refers to the version of SML/NJ installed on Andrew.
In the working version of SML/NJ the type of this function is:

val TextIO.inputLine: instream -> string option

so you would have to do things a bit differently if you use the
working version.

As an input stream we will use TextIO.stdIn.

The first problem is to turn the create a char stream that returns
the values coming from standard input.
*)

(* Include the streams from lecture *)
use "streams.sml";

structure S = Stream(structure BasicStream = BasicStream);

val stdin_stream: char S.stream = raise Unimplemented

(*
Hint: The basic idea is to read a string from the terminal and then
create a stream which returns one character at a time. When the string
has been fully consumed, a new string need to be read from the
terminal.
*)

(* Problem 2:

Now that you are able to read from the terminal, write a function that
prints a character stream to screen.

To do so you can use the TextIO.print function which has the following
type:

val TextIO.print : string -> unit
*)

val print_stream: char S.stream → unit = fn _ ⇒ raise Unimplemented

(*
Hint: You need to convert the characters into strings to print them.
*)
```

```
(* Problem 3:

Now you can put the two together to write a function where echos what
it receives from the console.
*)

val echo: unit → unit = fn _ ⇒ raise Unimplemented

(* Problem 4:

The last problem is not very interesting, because it just echos
character verbatim.

Now write a function called lamer_conv which converts a stream of
character in a new stream of character doing the following
substitutions:

s => 5
S => $
e => 3
E => 3
o => 0
O => 0
i => 1
I => 1
l => 1
L => 1
a => @
A => @
t => 7
T => 7

E.g., given a stream containing the characters in the string:

"This is a sentence"

should return a stream containing the characters in the string:

"7h15 15 @ 53n73nc3"
*)

val lamer_conv: char S.stream → char S.stream = fn _ ⇒ raise Unimplemented

(*
Hint: You want to do this without forcing the stream, i.e., you should
return the converted stream before the user actually input the text.
Look at the stream functions for the appropriate one.
*)

(* Problem 5:

Write now a different conversion function case_conv which capitalizes
every consonant of a char stream.

E.g., given a stream containing the characters in the string:

"This is a sentence"
should return a stream containing the characters in the string:

"THiS iS a SeNTeNCe"
*)

val case_conv: char S.stream → char S.stream = fn _ ⇒ raise Unimplemented

(* Problem 6:

Put together the result of the last two problems to write a function
crazy_echo which echoes the text received from the terminal with both
the conversions of problem 4 and 5.
*)

val crazy_echo: unit → unit = fn _ ⇒ raise Unimplemented

(* Problem 7:

Change the function in problems 4, 5, 6 to perform conversions
only on one character every n, where n is an argument to the
functions. You want crazy_echo_n to receive two (possibly different)
integers to pass to the two function lamer_conv_n and case_conv_n.
*)

val lamer_conv_n: int → char S.stream → char S.stream =
  fn _ ⇒ raise Unimplemented

val case_conv_n: int → char S.stream → char S.stream =
  fn _ ⇒ raise Unimplemented

val crazy_echo: (int * int) → unit → unit =
  fn _ ⇒ raise Unimplemented
```