



Tarea 1: Fundamentos

Asignatura: PROGRAMACION LOGICA Y FUNCIONAL

Profesor: DUARTE ACHACH, RAUL ALBERTO

E17081430 Novelo Cruz Raúl Armín

E17081478 Álvarez Pacheco Adolfo Esteban

E17081438 Herrera Garnica Fernando A. A.

E17081479 Esteban E. Brito Borges

E17081416 Paúl A. Mena Zapata

Carrera: Ingeniería en Sistemas Computacionales

Grupo: 7SA

Septiembre 2020

Test y política de la rueda

Nota: Los ejercicios marcados con (*) no se realizaron. Como señaló el profesor

Tarea 1. (1 punto) (*)

Debería haber recibido un mensaje de correo electrónico con instrucciones sobre cómo registrarse en Piazza. Activa tu cuenta. Hay una publicación anclada allí con una imagen. Describa brevemente esta imagen.

Tarea 2. (1 punto) (*)

¿Cuál es el comando terminal que se puede utilizar para descargar el primer laboratorio? ¿Desde qué directorio debe ejecutar este comando?

Tarea 3. (1 punto) (*)

¿Cuál es el comando terminal que se puede utilizar para copiar el archivo handin.zip, ubicado en AFS en sample@unix.andrew.cmu.edu:private/15150/basics, en el escritorio en su local

¿Máquina? Este comando debe ejecutarse desde el terminal local.

Tarea 4. (1 punto) (*)

¿Dónde puede encontrar la plantilla de LATEX para la tarea escrita de esta semana?

Tarea 5. (1 punto) (*)

Identifique uno de los editores de código sugeridos en la guía Great Practical Ideas for 150 en el sitio web 15-150.

Tarea 6. (1 punto)

Imagine que ha declarado una función que cumple con la siguiente especificación:

$\text{pow2} : \text{int} \rightarrow \text{int}$

REQUIERE: $n \geq 0$

ASEGURA: $\text{pow2 } n \cong 2^n$

Proporcione un caso de prueba de ejemplo para pow2 mediante una sintaxis válida.

Solución:

En la terminal, después de definir la función pow2, podemos escribir:

```
-def pow2 x = x * x;
```

```
-pow2 4
```

```
-val it = 16 : int
```

Con esto Podemos comprobar que pow2 funciona tal como se muestra en la definición anterior.

Lea la política de colaboración en el sitio web del curso. Para cada una de las siguientes situaciones, decida si las acciones de los estudiantes están permitidas o no por la política. Explica tus respuestas.

Tarea 7. (1 punto) (*)

Mia y Rahjshiba están almorzando juntos en Zoom. Mia menciona que había descubierto cómo resolver un problema específico. Rahjshiba, que no había pensado previamente en el problema, habla con Mia sobre su enfoque. A continuación, cada uno cierra la sesión y escribe la solución por separado.

Tarea 8. (1 punto) (*)

Harrison y James son amigos que toman 150 juntos. Durante la conferencia, James se confunde con uno de los ejemplos que se cubren. Le pregunta a Harrison después de clase, así que Harrison se lo explica.

Tarea 9. (1 punto) (*)

Edward está trabajando hasta tarde en una pregunta complicada y no puede entenderlo. Para obtener una pista, le da un mensaje a su amigo que también está tomando el curso y se va a la cama. A la mañana siguiente, lee las pistas de su amigo y trabaja la solución a partir de ahí.

Tarea 10. (1 punto) (*)

Soumil está atascado en un problema de la tarea y no está seguro de qué hacer. Comienza a leer un libro de texto de programación funcional que encontró en la biblioteca para pedir ayuda. En el libro, hay un problema de ejemplo que resuelve el problema en el que está atrapado. Luego lee su solución, y usa ese ejemplo para construir su respuesta. ¿Está permitido por la política? ¿Y si menciona en collab.txt que consultó el libro de texto?

Tarea 11. (1 punto) (*)

Tim y Kaz viven en la misma casa y ambos están tomando 150. Tim está trabajando en un problema solo en una pizarra en su sala de estar. Accidentalmente se olvida de borrar su solución y la escribe solo más tarde. Más tarde, Kaz, que había olvidado la asignación hasta el último día, pasa y ve la solución. Lo lee, lo borra y luego escribe su solución.

3 Tipos

Para compilar correctamente, un programa SML solo debe contener expresiones bien tipadas. Podemos documentar los tipos de expresiones que utilizamos en nuestros programas utilizando anotaciones de tipo, como en `15150 : int`. Sin embargo, SML realiza la comprobación automática de tipos mediante varias reglas de escritura, independientemente de si se incluyen estas anotaciones.

Una de estas reglas de escritura se refiere a las expresiones de aplicación. En un tipo de función como `t1 -> t2` (para algunos tipos `t1` y `t2`), `t1` es el tipo de argumento y `t2` es el tipo de resultado. Por lo tanto, una aplicación `e2 e1` está bien tipada si la expresión `e2` tiene un tipo de función `t1 -> t2` y la expresión de argumento `e1` tiene el tipo de argumento correcto `t1`. A continuación, la aplicación tiene el tipo de resultado correspondiente `t2`. Podemos escribir esta regla de escritura para la aplicación de la función (app abreviada) de la siguiente manera:

[APP] Si $e2 : t1 \rightarrow t2$ y $e1 : t1$, entonces $(e2\ e1) : t2$.

Por ejemplo, supongamos que `Int.toString` tiene el tipo `int -> string`. Considere la expresión de aplicación `Int.toString 7`. Ya hemos dicho que `Int.toString` tiene

el tipo `int` -> `string`, un tipo de función con el tipo de argumento `int` y la cadena de tipo de resultado. Claramente `7` tiene tipo `int`. Dado que se trata del tipo de argumento correcto, la aplicación `Int.toString 7` tiene la cadena de tipo de resultado correspondiente.

Podemos formalizar esta discusión de la siguiente manera:

1. `Int.toString : int -> string`
2. `7 : int`
3. `(Int.toString 7) : string` by [APP]

Tarea 12. (3 puntos)

Determine el tipo de expresión:

$(Int.toString\ 115) \wedge (Int.toString\ 35)$

Como vimos `Int.toString` tiene el tipo `int -> string`. Considerando la expresión de aplicación `Int.toString 115`, este argumento es de tipo `int` por lo que el argumento es correcto.

Solución.

The screenshot shows a web browser window with the address bar displaying `itmerida.brightspace.com/d2l/home/9188`. The page content includes a sidebar with 'SEP' and 'Contenido' sections, and a main area with a red header 'PR' and a 'My Cou' section. Overlaid on this is a Microsoft Word document titled 'tarea-1' by 'ADOLFO ESTEBAN ALVAREZ PACHECO'. The document contains the following text:

3. `(Int.toString 7): string` por [APP]

Tarea 12. (3 puntos)

Determine el tipo de expresión:

`(Int.toString 115) (Int.toString 35)`

- Como vimos `Int.toString` tiene el tipo `int -> string`. Considerando la expresión de aplicación `Int.toString 115`, este argumento es de tipo `int` por lo que el argumento es correcto.

Podemos formalizar esta discusión de la siguiente manera:

- `Int.toString: int -> string`
- `115: int`
- `(Int.toString 115): string` by [APP]

- Como ya se mencionó antes, `Int.toString` tiene el tipo `int -> string`. Considerando la expresión de aplicación `Int.toString 35`, este argumento es de tipo `int` por lo que el argumento es correcto.

Podemos formalizar esta discusión de la siguiente manera:

- `Int.toString: int -> string`
- `35: int`
- `(Int.toString 35): string` by [APP]

The Word document footer shows 'Página 3 de 11', '2746 palabras', and a 'Concentración' button.

Podemos formalizar esta discusión de la siguiente manera:

Int.toString: int -> string

115: int

(Int.toString 115): string by [APP]

Como ya se mencionó antes, Int.toString tiene el tipo int -> string. Considerando la expresión de aplicación Int.toString 35, este argumento es de tipo int por lo que el argumento es correcto.

Podemos formalizar esta discusión de la siguiente manera:

Int.toString: int -> string

35: int

(Int.toString 35): string by [APP]

Describe su razonamiento de la misma manera que lo anterior, primero utilizando informalmente el inglés, luego resume usando la notación más formal. Si parte de su razonamiento corresponde exactamente al que se encuentra en el ejemplo, no dude en citar la correspondencia en lugar de copiar todo.

Tarea 13. (2 puntos)

Explicar por qué la expresión Int.toString 2.0 no está bien tipada.

Solución.

The screenshot shows a web browser window with the URL `itmerida.brightspace.com/d2l/home/9188`. The page title is "PROGRAMACION LOGICA ...". Below the browser window, there is a Microsoft Word document titled "Tarea 13. (2 puntos)". The document content is as follows:

Tarea 13. (2 puntos)
Explicar por qué la expresión `Int.toString 2.0` no está bien tipada.

- Como ya se mencionó en repetidas ocasiones, `Int.toString` tiene el tipo `int -> string`. Considerando la expresión de aplicación `Int.toString 2.0`, este argumento NO es de tipo `int` por lo que el argumento es incorrecto.

Por lo que se concluye con:

- `Int.toString: int -> string`
- `2.0: real`
- `(Int.toString 2.0): Error: operator and operand do not agree [tycon mismatch]`

The Word document also shows a status bar at the bottom indicating "Página 4 de 12", "2746 palabras", and "Concentración".

Como ya se mencionó en repetidas ocasiones, `Int.toString` tiene el tipo `int -> string`. Considerando la expresión de aplicación `Int.toString 2.0`, este argumento NO es de tipo `int` por lo que el argumento es incorrecto.

Por lo que se concluye con:

`Int.toString: int -> string`

`2.0: real`

`(Int.toString 2.0): Error: operator and operand do not agree [tycon mismatch]`

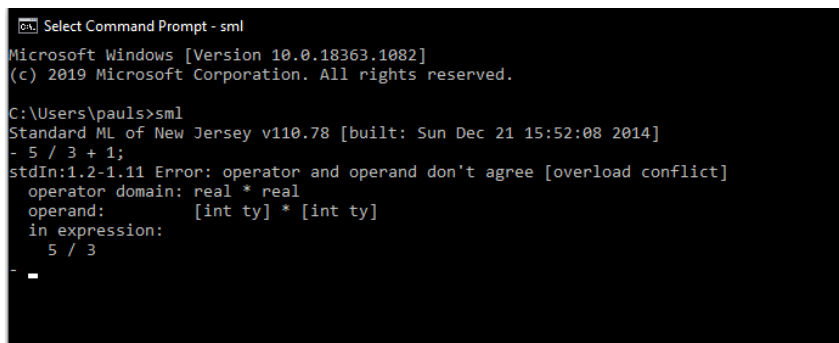
Para cada una de las siguientes expresiones, indique su tipo. No es necesario proporcionar razonamiento adicional. Si no está bien tipado, ponga "no bien escrito" (NWT). Debido a que el razonamiento sobre los tipos es una habilidad importante para el curso, asegúrese de que tiene una comprensión completa de las siguientes tareas y evite el uso de la REPL SML/NJ.

Tarea 14. (1 punto)

$5 / 3 + 1$

Solución.

ES DE TIPO DE DATO REAL



```
Select Command Prompt - sml
Microsoft Windows [Version 10.0.18363.1082]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\pauls>sml
Standard ML of New Jersey v110.78 [built: Sun Dec 21 15:52:08 2014]
- 5 / 3 + 1;
stdIn:1.2-1.11 Error: operator and operand don't agree [overload conflict]
  operator domain: real * real
  operand:         [int ty] * [int ty]
  in expression:
    5 / 3
-
```

Tarea 15. (1 punto)

(fn x > x + 1)

Solución.

ES DE TIPO ENTERO (INT)

```
Command Prompt - sml

C:\Users\pauls>sml
Standard ML of New Jersey v110.78 [built: Sun Dec 21 15:52:08 2014]
- (fn x => x + 1)
= ;
val it = fn : int -> int
-
```

Tarea 16. (1 punto)

(2 + 4, 8.0)

Solución.

TUPLA CON VALORES ENTERO Y REAL

```
Command Prompt - sml

C:\Users\pauls>sml
Standard ML of New Jersey v110.78 [built: Sun Dec 21 15:52:08 2014]
- (2 + 4, 8.0)
= ;
val it = (6,8.0) : int * real
-
```

Tarea 17. (1 punto)

"15" ^ "150"

Solución.

UNION DE STRINGS QUE DA UN STRING

```
Command Prompt - sml

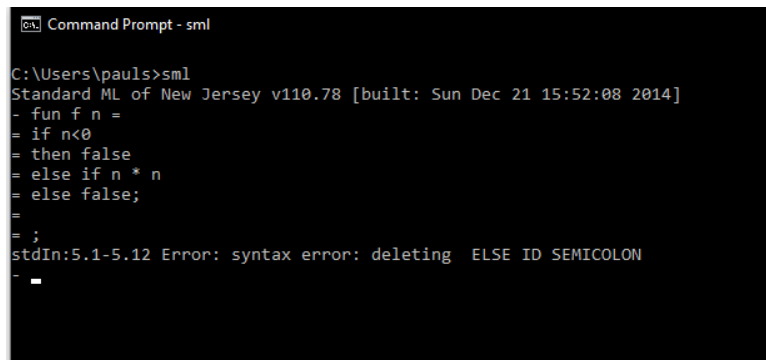
C:\Users\pauls>sml
Standard ML of New Jersey v110.78 [built: Sun Dec 21 15:52:08 2014]
- "15" ^ "150"
= ;
val it = "15150" : string
-
```


Tarea 18. (1 punto)

diversión $f\ n$ si $n < 0$ entonces false else $n * n$

Solución.

FUNCION QUE RECIBE ENTERO Y DEVUELVE BOOLEANO O ENTERO,
PERO DA ERROR "TYPES OF IF BRANCHES DO NOT AGREE"



```
Command Prompt - sml
C:\Users\pauls>sml
Standard ML of New Jersey v110.78 [built: Sun Dec 21 15:52:08 2014]
- fun f n =
= if n<0
= then false
= else if n * n
= else false;
=
= ;
stdIn:5.1-5.12 Error: syntax error: deleting ELSE ID SEMICOLON
-
```

4 Evaluación

Se puede evaluar una expresión bien tipada. Si su evaluación finaliza sin generar una excepción, el resultado es un valor. Si la expresión ya es un valor como un número entero o una función (FUNCTIONS ARE VALUES), no se evalúa más. En una expresión como e_1 a e_2 , el operador de concatenación de infijos $\dot{\wedge}$ evalúa sus dos argumentos, e_1 y e_2 , de izquierda a derecha y, a continuación, devuelve la cadena obtenida concatenando las dos cadenas que resultan de estas evaluaciones.

A continuación, se muestra un ejemplo: considere la expresión $(\text{Int.toString } 7) \dot{\wedge} "1"$. Supongamos que la aplicación $\text{Int.toString } 7$ se evalúa como el valor "7". La expresión "1" ya es un valor. Por lo tanto, la expresión $(\text{Int.toString } 7) \dot{\wedge} "1"$ se evalúa como "71", la cadena creada concatenando "7" y "1".

Usando la notación de la clase, escribimos $e \Rightarrow e_0$ cuando e se reduce a e_0 en un número finito de pasos (cuando una expresión "se reduce a" un valor que

también podemos decir "evalúa a"). Podemos resumir los hechos relevantes sobre la evaluación en este ejemplo como:

`(Int.toString 7) ^ "1"`

• \Rightarrow `"7" ^ "1"`

\Rightarrow `"71"`

Ahora le pedimos que realice un análisis similar en otro ejemplo. Supongamos que la expresión `fact 4` se evalúa como `24` y que la función `Int.toString` tiene el comportamiento habitual, por ejemplo, `Int.toString 150` se evalúa como `"150"`.

Tarea 19. (2 puntos)

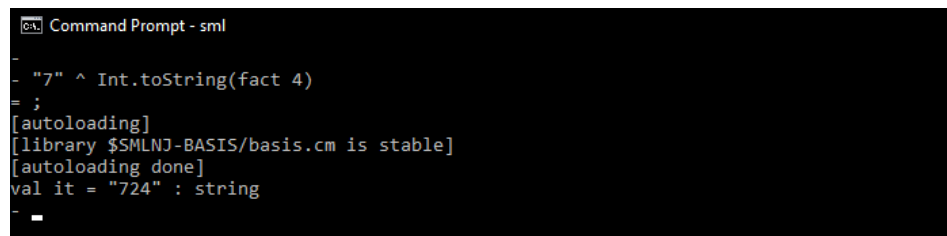
Determine el valor resultante de la siguiente expresión:

`"7" ^ Int.toString (fact 4)`

Explique su razonamiento informalmente de la misma manera que lo anterior.

Solución.

EL RESULTADO ES AMBOS CONCATENADOS COMO STRING = `"714"`



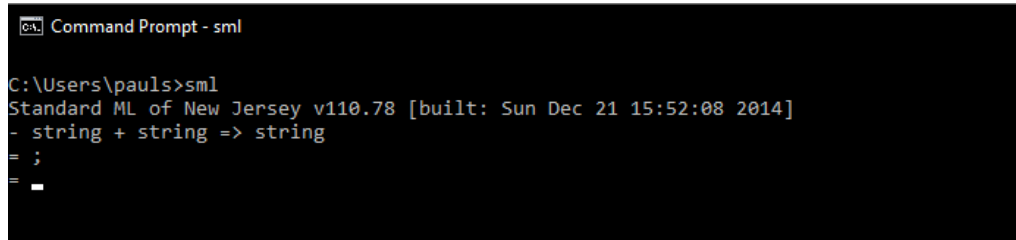
```
Command Prompt - sml
> "7" ^ Int.toString(fact 4)
= ;
[autoloading]
[library $SMLNJ-BASIS/basis.cm is stable]
[autoloading done]
val it = "724" : string
>
```

Tarea 20. (3 puntos)

Ahora utilice la notación de la clase \Rightarrow de la clase, como se ha indicado anteriormente, para expresar los hechos clave de la evaluación en el análisis.

Solución.

string + string \Rightarrow string



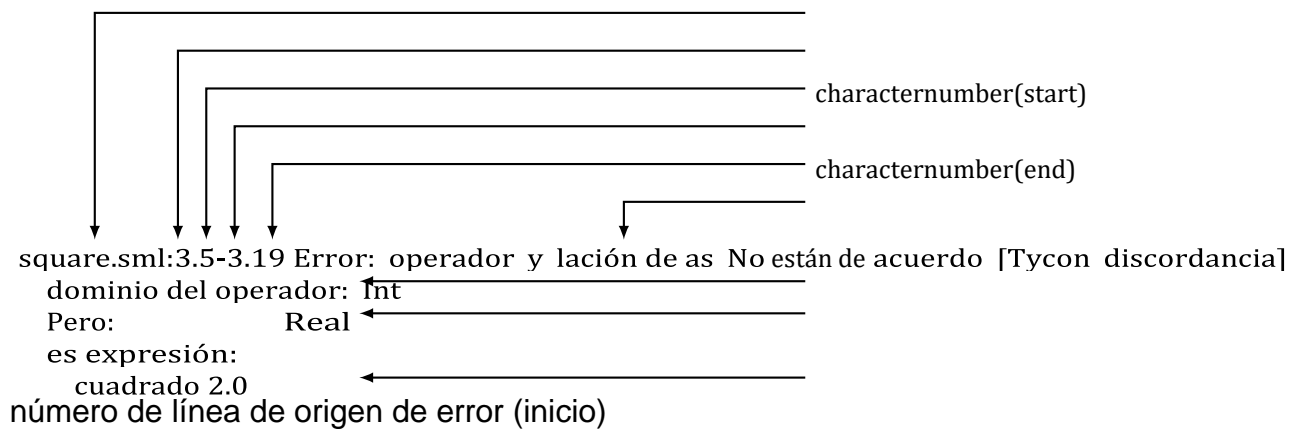
5 mensajes de error

5.1 Leer mensajes de error

Supongamos que teníamos el siguiente código en un archivo llamado square.sml:

cuadrado de la diversión (n : int) : int á n * n resultado val á cuadrado 2.0

Si intentamos compilarlo escribiendo `smlnj square.sml` en nuestro terminal, SML / NJ nos da un mensaje de error! A continuación, le explicamos cómo leerlo:



número de línea (fin)

nombre del error

tipo SML tipo esperado de la expresión de código para corregir

Para solucionarlo, podríamos cambiar el 2.0 a 2.

5.2 Interpretar mensajes de error

Para esta tarea, usaremos el archivo `code/errors/errors.sml`. Antes de comenzar, `cd` en el directorio `code/errors/`. directory.

Puede evaluar las declaraciones SML en este archivo utilizando el comando `use "errors.sml";`

dentro de la REPL SML/NJ. Alternativamente, puede ejecutar el archivo directamente a través de `smlnj errors.sml`

Si no se encuentra el comando `smlnj`, consulte el folleto Lab 1 para obtener instrucciones sobre cómo configurar su `PATH`.

Desafortunadamente, el archivo tiene algunos errores que deben corregirse. Las siguientes cinco tareas le guiarán a través del proceso de corrección de estos errores. Asegúrese de enviar respuestas por escrito a Gradescope para recibir el crédito completo.

Tarea 21. (2 puntos) (*)

¿Qué mensaje de error aparece al evaluar el archivo `code/errors/errors.sml` no modificado? ¿Qué causó este error? ¿Cómo se puede arreglar?

Nota: ¡No es necesario reescribir el cuerpo de la función!

Corrija este error en el código de archivo `code/errors/errors.sml` y evalúe el archivo de nuevo utilizando el mismo comando que antes.

Tarea 22. (2 puntos) (*)

Con el primer error corregido, se encontrará con un conjunto de errores. ¿Cuál es el primer error de este conjunto? ¿Qué causó este error? ¿Cómo lo arreglas? ? Tenga en cuenta que estos errores deben referencia diferentes líneas que el primer error.

Corrija este error en el código de archivo `code/errors/errors.sml` y vuelva a evaluar el archivo.

Tarea 23. (2 puntos) (*)

Debería ver más mensajes de error. ¿Cuáles son los dos primeros mensajes de error? Ambos deben hacer referencia a la misma línea de código SML. ¿Qué significan estos mensajes de error? ¿Cómo los arreglas?

Ambos errores deben desaparecer con una corrección. Corrija los errores y vuelva a evaluar el archivo.

Tarea 24. (2 puntos) (*)

Ahora hay otro conjunto de errores. ¿Cuál es el primer mensaje de error que ve ahora? ¿Qué significa este mensaje de error? ¿Cómo se soluciona este error?

Una vez más, corrija el error y, a continuación, vuelva a evaluar el archivo.

Tarea 25. (2 puntos) (*)

Debería haber un mensaje de error más. ¿Qué es? ¿Qué lo causó? ¿Cómo se soluciona este error?

Tarea 26. (5 puntos) (*)

Cuando corrija este error final y evalúe el archivo no debería haber más mensajes de error. Envíe el código a Gradescope y compruebe que se compila limpiamente en Gradescope.

Tarea 27. (0 puntos) (*)

¡Aunque esta tarea vale cero puntos, recibirá puntos negativos en caso de que no pueda completarla! Observe la salida de Gradescope; debe notar que el calificador encontró un error de estilo en el código enviado. Corrija la infracción de estilo y envíela de nuevo a Gradescope, comprobando que no se ha perdido ningún punto de estilo.

6 valores esperados y funciones

Considere la siguiente función:

```
(* decimal : int -> int list *)  
fun decimal (n : int) : int list =  
    if n < 10 then [n] else (n mod 10) :: decimal (n div 10)
```

Una especificación para esta función tiene la forma típica

```
decimal : int -> Int Lista
```

REQUIERE: . . .

Asegura:

La función satisface esta especificación si para todos los valores n de tipo `int` que satisfacen la propuesta descrita en `Requiere`, la evaluación de `decimal n` satisface la proposición descrita en `Aseguras`.

Para cada una de las especificaciones siguientes, diga si esta función cumple o no la especificación. Si no es así, dé un ejemplo para ilustrar lo que sale mal.

Nota: Un dígito (decimal) es un valor entero en el rango 0 - 9.

Tarea 28. (2 puntos)

```
decimal : int -> int list
```

REQUIERE: $n > 0$

ENSURES: `decimal n` se evalúa como una lista no vacía de dígitos

Solución.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\Raul Novelo\Dropbox\7SA (1)\ACHACH> sml .\basics\28-32.sml
Standard ML of New Jersey (32-bit) v110.98.1 [built: Wed Aug 26 08:41:54 2020]
[opening .\basics\28-32.sml]
val decimal = fn : int -> int list

- decimal(0);
val it = [0] : int list

- █
```

Tarea 29. (2 puntos)

decimal : int -> int list

REQUIERE: $n \geq 0$

ENSURES: decimal n se evalúa como una lista no vacía de dígitos

Solución.

```
PS C:\Users\Raul Novelo\Dropbox\7SA (1)\ACHACH> sml .\basics\28-32.sml
Standard ML of New Jersey (32-bit) v110.98.1 [built: Wed Aug 26 08:41:54 2020]
[opening .\basics\28-32.sml]
val decimal = fn : int -> int list

- decimal(10);
val it = [0,1] : int list

- decimal(0);
val it = [0] : int list

- decimal(100);
val it = [0,0,1] : int list

- decimal(126);
val it = [6,2,1] : int list
```

Tarea 30. (2 puntos)

decimal : int -> int list

REQUIERE: true

ENSURES: decimal n se evalúa como una lista no vacía de dígitos

Solución

```
- decimal(true);
stdIn:2.1-2.14 Error: operator and operand do not agree [tycon mismatch]
operator domain: int
operand:         bool
in expression:
  decimal true

- 
```

No tiene ningún sentido recibir un parámetro booleano ya que está pidiendo un int

Tarea 31. (2 puntos)

decimal : int -> int list

REQUIERE: $n \geq 0$

ENSURES: decimal n se evalúa como una lista de dígitos

Solución.

```
- decimal();
stdIn:1.2-1.11 Error: operator and operand do not agree [tycon mismatch]
  operator domain: int
  operand:         unit
  in expression:
    decimal ()
- |
```

Para cualquier número $n \geq 0$ siempre devuelve una lista y esta

Tarea 32. (2 puntos)

¿Cuál de estas especificaciones proporciona la información más útil sobre el comportamiento de la función decimal? Di por qué, brevemente.

Solución.

La segunda opción, debido a que expresa de manera correcta la evaluación del parámetro N que las demás dejaban muy ambiguo. Al probar la función nos damos cuenta de que N es mayor o igual que cero y que para cualquier número incluso el cero siempre devolverá una lista de al menos 1 elemento en ella.

decimal : int -> int list

REQUIERE: $n \geq 0$

ENSURES: decimal n se evalúa como una lista no vacía de dígitos

7Scopio

Recuerde desde el laboratorio que decimos que una declaración está dentro del ámbito de un enlace si podemos usar ese enlace para hacer esa declaración. Considere el siguiente ejemplo:

```
val x : int = 3
val y : int = x + 1
val x : int = 10
val z : int = x + 1
```

Decimos que, `y` está en el ámbito del primer enlace de `x`, pero no en el ámbito del segundo enlace de `x` (porque se creó después de que `y` fue enlazado), por lo que `y` se enlaza a `4`. Podemos decir que `z` está en el ámbito de la segunda unión de `x`; `z` NO está en el ámbito del primer enlace de `x` porque el segundo enlace sombrea el primero, por lo que `z` se enlaza a `11`. Para cualquier identificador enlazado varias veces, las nuevas declaraciones solo están en el ámbito del enlace más reciente de ese identificador. La función incorporada

`real : int -> real`

devuelve el valor real correspondiente a una entrada `int` determinada; por ejemplo, `real 1` se evalúa como

1.0. Por el contrario, la función integrada

`trunc : real -> int`

devuelve la parte integral (intuitivamente, los dígitos antes del punto decimal) de su entrada; por ejemplo, `trunc 3.9` se evalúa como `3`. No dude en probar estas funciones en el SML/NJ REPL.

Una vez que entienda estas funciones, debe resolver las preguntas de esta sección en su cabeza, sin probarlas primero en el REPL SML/NJ. Recuerde, no tendrá acceso a SML / NJ durante un examen, por lo que debe ser capaz de razonar sobre el código a mano.

Tarea 33. (3 puntos)

Considere el siguiente fragmento de código:

```
fun squareit (a : real) : real = a * a
fun squareit (b : real) : int = trunc b * trunc b
fun bopit     (c : real) : real = squareit (c + 1.0)
```

¿Este tipo de control? Explicar brevemente por qué o por qué no.

Solución.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Users\Raul Novelo\Dropbox\7SA (1)\ACHACH> sm1 .\basics\33.sml
Standard ML of New Jersey (32-bit) v110.98.1 [built: Wed Aug 26 08:41:54 2020]
[opening .\basics\33.sml]
val squareit = fn : real -> real

val squareit = fn : real -> int

.\basics\33.sml:3.5-3.47 Error: right-hand-side of clause does not agree with function result type [tycon mismatch]
expression:  int
result type:  real
in declaration:
    bopit = (fn c : real => squareit (c + 1.0): real)

PS C:\Users\Raul Novelo\Dropbox\7SA (1)\ACHACH> █
```

Tarea 34. (6 puntos)

Considere el siguiente fragmento de código:

```
val x : int = 3
fun foo (w : int, x : int) : int = 2 + x
val y : int = x
val x : int = 4
val z : int = foo (y, x)
```

¿A qué valor se enlaza w cuando se llama a foo en la línea 5? y ¿Por qué? Por la posición de los argumentos

Solución.

```
PS C:\Users\Raul Novelo\Dropbox\7SA (1)\ACHACH> sml .\basics\34.sml
Standard ML of New Jersey (32-bit) v110.98.1 [built: Wed Aug 26 08:41:54 2020]
[opening .\basics\34.sml]
val x = 3 : int

val foo = fn : int * int -> int

val y = 3 : int

val x = 4 : int

val z = 6 : int
```

Esta asignación tiene un total de 60 puntos.