Trabajo inicial.

I. Ejercicios sml.

A continuación unos ejercicios de sml. Son cinco. Trata de realizarlos todos. Cada ejercicio, menos el primero, hace uso del anterior.

1. Escriba una función is_older función que toma dos fechas y se evalúa como verdadera o falsa. Se evalúa a verdadero si el primer argumento es una fecha que viene antes del segundo argumento . (Si las dos fechas son iguales, el resultado es falso.)

2. Escriba una función number_in_month que toma una lista de fechas y un mes (es decir, un int) y devuelve el número de fechas en la lista que contienen el mes determinado.

3. Escriba una función number_in_months que toma una lista de fechas y una lista de meses (es decir, una lista int) y devuelve el número de fechas en la lista de las fechas que se encuentran en alguno de los meses en la lista de meses. Supondremos que la lista de los meses no hay ha repetido ningún número. Sugerencia: Utilice su respuesta al problema anterior.

4. Escriba un función dates_in_month que toma una lista de fechas y un mes (es decir, un int) y devuelve una lista que contiene las fechas de la lista de argumentos de fechas que están en el mes. La lista devuelta debe contener las fechas en el orden que se les dio en un principio.

5. Escriba una función dates_in_months que toma una lista de fechas y una lista de meses (es decir, una lista int) y devuelve una lista que sostiene las fechas de la lista de argumentos de las fechas que se encuentran en alguno de los meses en la lista de meses. Supondremos que en la lista de los meses no hay repetido ningún número. Sugerencia: Utilice su respuesta al problema anterior y el operador de lista append (@).

II.- Más ejercicios SML

1.- alternate

Escriba una función alternate: int list -> int que tome una lista de números y los agregue con signo alterno. Por ejemplo, suplente [1,2,3,4] = 1 - 2 + 3 - 4 = -2.

2.- min_max

Escriba una función min_max : int list -> int * int que tome una lista de números no vacía y devuelva un par (min, max) del mínimo y el máximo de los números de la lista.

3.- cumsum

Escriba una función cumsum: int list -> int list que tome una lista de números y devuelva una lista de las sumas parciales de esos números. Por ejemplo cumsum [1,4,20] = [1,5,25].

4.- saludo

Escriba un función saludo : opción de cadena -> cadena que dada una opción de cadena SOME nombre devuelve la cadena "¡Hola, ...!" donde los puntos serían reemplazados por nombre. Tenga en cuenta que el nombre se proporciona como una opción, por lo que si es None, reemplace los puntos con "usted".

5.- repetir

Escriba una función repetir: list int * lista int -> lista int que dada una lista de enteros y otra lista de enteros no negativos, repite los enteros en la primera lista de acuerdo con los números indicados por la segunda lista. Por ejemplo: repetir ([1,2,3], [4,0,3]) = [1,1,1,1,3,3,3].

6.- addOpt

Escriba una función addOpt: opción int * opción int -> optión int que, dados dos enteros "opcionales", los agrega si ambos están presentes (devolviendo SOME de su suma), o devuelve NONE si al menos uno de los dos argumentos es NONE .

7.- agregarTodasOpt

Escriba una función agregarTodasOpt: int option list -> int option que, dada una lista de enteros "opcionales", agrega los enteros que están allí (es decir, agrega todos los SOME i). Por ejemplo: addAllOpt ([SOME 1, NONE, SOME 3]) = SOME 4. Si la lista no contiene SOME i, es decir, todos son NONE o la lista está vacía, la función debería devolver NONE.

8.- cualquiera

Escribe una función cualquiera : bool list -> bool que dada una lista de valores booleanos devuelve verdadero si hay al menos uno de ellos que es verdadero, de lo contrario devuelve falso. (Si la lista está vacía, debería devolver falso porque no hay verdadero).

9.- todo

Escribe una función all : bool list -> bool que dada una lista de valores booleanos devuelve verdadero si todos son verdaderos, de lo contrario devuelve falso. (Si la lista está vacía, debería devolver verdadero porque no hay falso).

10.- zip

Escribe una función zip: int list * int list -> int * int que dadas dos listas de enteros crea pares consecutivos y se detiene cuando una de las listas está vacía. Por ejemplo: zip ([1,2,3], [4, 6]) = [(1,4), (2,6)].