**CSE 341 -- Programming Languages**

**Autumn 2003**

**Department of Computer Science and Engineering, University of Washington**

**Steve Tanimoto (instructor)**

# Assignment 5

# Next in a Sequence (ML)

Due date and time: **Wednesday, November 10: Web-based turn-in at 11:59 PM.)** .

Version 1.0 of 5 November

## Title: Next in a Sequence (ML).

**Purposes**: Gain fluency with ML and its style of functional programming.

**Team Format: Partnership Work -- Teams of 2**: for this assignment have been made by the INFACT group-formation algorithm, using any preference information you have entered in INFACT for who you want to work with. (You'll have another opportunity to express your preferences before the next group assignment.) Here are the groupings.
Each group should create and turn in one solution to the problem.

**Instructions**: Write a program in ML that analyzes sequences.

Your program should consist of several functions definitions. One of the functions you should have is one with the name **next**. Here is an example of using **next**.

```
next [1,3,5];
val it = 7 : int
next [~2, 10, ~50];
val it = 250 : int
next [0,10,0,11,0,12,0];
val it = 13 : int
```

Another function you should have is **findRecog(k:int, lst: int list): int -> int option**. This function takes a depth limit, k, and a list of integers, and uses a technique called breadth-first search to try to find a function f(n) that generates the given sequence. The sequence must be a "beta sequence" described below. If it finds such a function within the number of levels given by k, then it returns

SOME f, where f is the function found. Otherwise it returns NONE. (For more information on options, see Chapter 4 in Ullman.)

Another function you should implement is **gen**, used as follows:

```
fun f(n) = 7 + n*2;
val f = fn : int -> int

gen(f, 10);
val it = [7, 9, 11, 13, 15, 17, 19, 21, 23, 25] : int list
```

Or, putting findRecog and gen together, we could do the following:

```
- findRecog(5, [10,9,8]);
val it = SOME fn : fnopt
- gen(valOf(it),15);
val it = [10,9,8,7,6,5,4,3,2,1,0,~1,~2,~3,~4] : int list
```

As you can see, **gen** takes a function and an integer m, and it generates the first m terms in the sequence represented by the function. Here, the function is part of the function option returned by findRecog. findRecog has searched for and found a function that generates the arithmetic sequence starting with 10, 9, and 8, and it has returned the function option SOME with the desired function. The accessor, valOf, applied to the function option, returns the function itself.

A *beta sequence* is defined recursively as follows.

- Any arithmetic sequence of integers is a beta sequence. For example 2, 4, 6, ... is a beta sequence.
- Any geometric sequence of integers is a beta sequence. For example 2, 6, 18, 54, ... is a beta sequence.
- Any "interleaved" sequence C made from two beta sequences A and B is a beta sequence. This means that the even-index elements of C (starting with index 0) come from A and the odd-index elements of C come from B. That is, C has the form

  ```
  C = [a0, b0, a1, b1, ...]
  where
  A = [a0, a1, a2, ...]
  and
  B = [b0, b1, b2, ...]
  ```

For up to 5 points of extra credit, implement the following additional function (with any appropriate helper functions): **findDesc**. This function is analogous to findRecog, except that instead of returning a function option, it returns a string. The string describes how the sequence is generated. The description provides three parts: 1. the type of sequence (A.S. = Arithmetic Sequence, G.S. = Geometric Sequence, and I.S. = Interleaved Sequence); 2. a function generating the sequence in the case of A.S. or G.S., and in the case of I.S., a description of the even and odd subsequences (done recursively); and 3. if it is an A.S. or a G.S, then a list showing the first 3 terms of the sequence.

If findDesc is not successful in finding the correct description of the sequence within the given search depth, it should return the string "None". This way, we do not need to use a string option as the return type of this function.

```
- next [3,5,7];
val it = 9 : int
```

```
- findDesc(10, [3,5,7]);
val it = "(A.S. f(n)=3+n*2:[3,5,7,...])" : string
- findDesc(10, [~2, 10, ~50]);
val it = "(G.S. f(n)=~2*(~5)^n:[~2,10,~50,...])" : string
- next [1, 10, 2, 10];
val it = 3 : int
- findDesc(10, [1, 10, 2, 10]);
val it =
  "(I.S. of (A.S. f(n)=1+n*1:[1,2,3,...]) with (A.S. f(n)=10+n*0:[10,10,10,...]))"
  : string
- next [1,2,5,43,6];
val it = 84 : int
- findDesc(10, [1,2,5,43,6]);
val it =
  "(I.S. of (I.S. of (A.S. f(n)=1+n*5:[1,6,11,...]) with (A.S. f(n)=5: [5,5,5,...]
)) with (A.S. f(n)=2+n*41:[2,43,84,...]))"
  : string
- findDesc(2, [1,2,5,43,6]);
val it = "None" : string
```

Evaluation: Part II is worth 30 points (not counting extra credit), with 25 for a working program that handles all the test cases and 5 points for style: clear design and implementation, and good comments.

**Helpful hints:**:
Use a new type identifier for your function options:

type fnopt = (int -> int) option;

To get SML to print the entire descriptions, and to suppress garbage collection messages, you should include the following in your program:

```
Compiler.Control.Print.printDepth := 1000;
Compiler.Control.Print.printLength := 1000;
Compiler.Control.Print.stringDepth := 1000;
SMLofNJ.Internals.GC.messages false;
```

Breadth-first search is a search technique that searches a tree of possibilities in order of increasing distance from the root. In creating your sequence recognizer, findRecog, you can implement breadth-first search very simply using essentially a depth-first recursive search in which you always check first for the given sequence being arithmetic, second for it being geometric, and only third being an interleaved sequence. Your test for an interleaved sequence should be another function findInterleavedRecog that also takes two arguments (the same kinds as findRecog) and that calls findRecog on each of the two subsequences obtained by splitting the input list into even and odd-indexed elements. The two functions findRecog and findInterleavedRecog should be co-recursive; you should use the "and" keyword of ML to join their definitions.

If you need another hint, you can peek at pseudocode for the central functions of the solution.