Problem 1 (10 pts.):
Suppose that we represent a polynomial as an SML list of coefficients, from lowest degree to highest degree. For example,

$$4x^3 + 2x^2 + 31$$

would be represented by the SML list,

[31.0, 0.0, 2.0, 4.0]

The length of the list will always be *n* + 1, where *n* is the degree of the polynomial.

For this problem please do the following:

A) (2 pts.)
Write an SML function named *genPoly(n)* that creates a polynomial of degree *n*, whose coefficients are all set to 1.0

Example: genPoly(3) = [1.0, 1.0, 1.0, 1.0]

B) (3 pts.)
Write an SML function called *evalPoly(P, a)* that takes a list representing a polynomial, *P*, and a real value, *a*, and returns *P*(*a*).

```
Example: evalPoly([10.0, 3.0, 1.0], 2.0) = 10 + 3.0(2.0) + 2.0^2
                                         = 10.0 + 6.0 + 4.0
                                         = 20.0
```

C) (5 pts.)
Write an SML function called *multPoly(P1, P2)* that takes two lists representing polynomials, multiplies the polynomials together, and returns a list representing their polynomial product.

Example: multPoly([~1.0, 1.0], [1.0, 1.0]) = (x - 1)(x + 1)
= x^2 + x - x - 1.0
= x^2 - 1.0
= [~1.0, 0.0, 1.0]

Please demonstrate the function written for each part of Problem 1 with at least two different inputs per function.

Problem 2 (10 pts. total):

Part A (4 pts.):
Write an SML function named *revCycle(L)* that takes as input a list, and then cycles the list once in reverse. That is, given a list

[a1, a2, a3, …, a_n]

your function returns the list

[a2, a3, …, a_n, a1].

Part B (6 pts.):
Write a recursive SML function named revCycles(L, i) that takes two inputs, an integer, 'i', and a list, 'l'. The function will produce a list that has been reverse cycled i times.

For example, if the inputs are:
[1, 2, 3, 4, 5, 6], 4

The list produced by the function should be:

[5, 6, 1, 2, 3, 4]

You must demonstrate the functions defined in parts A and B on at least 2 non-empty lists.

Problem 3 (10 pts.):

A) (5 pts)
Write an SML function called removeFst(x, L) that removes only the first instance of the element *x* from the list *L*. If *x* is not present then *L* is returned unchanged.

B) (5 pts)
Write an SML function called removeLst(x, L) that removes only the last instance of the element *x* from the list *L*. If *x* is not present then *L* is returned unchanged.

You must demonstrate the use of your functions from parts (A) and (B) on at least 2 non-empty lists each in your program.

Problem 4 (10 pts):

Write a set of mutually recursive functions that can be used to split a list into four(4) separate lists.

You can define more than two mutually recursive functions at a time linked by the "and" keyword. Each function in the set only has to call one of the other functions in the set for the entire set to be mutually recursive.

You must demonstrate the use of your function to split a list into four distinct lists in your program.

Problem 5 (10 pts):

Write a higher-order function, tabulate(a, d, n, F), which takes real values, *a* and *d*, an integer, *n*, and a function from reals to reals, *F*. Your function should return a list of tuples consisting of values *x* and *F*(*x*),
where x = *a*, *a* + *d*, *a* + 2*d*, …, *a* + *nd*

Example: tabulate(0.1, 2.0, 2, Square)
= [(0.1, Square(0.1)),(2.1, Square(2.1),(4.1, Square(4.1))]
= [(0.1, 0.01),(2.1, 4.41),(4.1, 16.81)]

You must demonstrate your tabulate function for at least 2 sets of inputs in your program.

Submission Instructions:

1. Please write each program in a separate file.
2. The file name should correspond to the problem for which your program is written.
   Example: Your program for problem 1 should be called "hw2p1.sml".
3. Please collect and submit your program files as a compressed zip file.