# CSE 341, Spring 2004, Assignment 2
# Due: Friday 16 April, 9:00AM

Last updated: 2 April

You will write seven functions and one type definition, in addition to using several datatype and type definitions we give you. One part of this assignment has to do with poker chips; the other part has to do with moving around a plane (the kind from math with an x-axis and a y-axis). No knowledge about chips is assumed. A little geometry is assumed, but just ask if something is unclear.

Your solutions must use pattern-matching. You may not use the functions `null`, `hd`, or `tl`, nor may you use anything containing a `#` character. You may not use mutation. Style matters.

1. A *chip* is one of three colors, as this binding suggests:

```
datatype chip = Blue | Red | White
```

   One way to represent a "chip collection" is as a list of chips. Another way is with a data structure that records only "how many of each color" are in the collection. Each color has a *cost*: 25 for Blue, 5 for Red, 1 for White. The cost of a collection is the sum of the costs of the chips in the collection.

   (a) Define a type `chip_summary` that represents a chip collection with the "how many of each color approach". Use a record type.

   (b) Define a function `chip_list_to_summary` that takes a list of chips and evaluates to the `chip_summary` that represents the same collection.

   (c) Define a function `summary_to_chip_list` that takes a `chip_summary` and evaluates to a list of chips that represents the same collection.

   (d) Define a function `chip_list_value_1` that takes a list of chips and evaluates to the list's cost. Your solution must not use any other functions.

   (e) Define a function `chip_list_value_2` that takes a list of chips and evaluates to the list's cost. Use `chip_list_to_summary` in your definition.

   *Food for thought: Why does part (b) say "evaluates to the" whereas part (c) says "evaluates to a"?*

2. This part uses these bindings:

```
type pos = real * real
type radians = real
datatype dir = Left | Right
datatype move = Forward of real | Turn of dir * radians | Home
```

   Assume the usual $x$,$y$ coordinate system with origin $(0.0, 0.0)$. A `pos` represents a position on the plane; the first component is the $x$-coordinate. There are $2\pi$ radians in a circle. Moves have this meaning:

   - `Forward d`: "go in the direction you are facing a *distance* of $d$" (A negative $d$ means backwards.)
   - `Turn(Left,r)`: "starting from the direction you are facing, rotate counter-clockwise r radians"
   - `Turn(Right,r)`: "starting from the direction you are facing, rotate clockwise r radians" (A negative number means rotate the other way.)
   - `Home`: "go to position $(0.0, 0.0)$ and face "*East*" (0.0 radians)"

   Note: ML does not implicitly convert from `int` to `real` (i.e., floating-point numbers). Write 0.0, not 0, for example.

   (a) Define a function `advance` that takes 3 arguments: A "current position" of type `pos`, an "angle" of type `radians`, and a "distance" of type `real` and returns the position one reaches by going length "distance" at angle "angle" starting at "current position". Hints:

- In geometry, $\Delta x = d\cos\theta$ and $\Delta y = d\sin\theta$.
- In ML, the library functions `Math.cos` and `Math.sin` take a value in radians and evaluate to what you think.

(b) Define a function `end_point` that takes a list of moves and returns the position one would end up at after following the moves. Assume you start at the origin facing East. Hints:
- Use another (possibly local) function.
- To turn Left, you add radians. To turn Right, you subtract radians.

(c) Define a function `visit` that takes a list of positions and evaluates to a list of moves. Starting at the origin and executing the moves would cause one to visit the list of positions in order. Hints:
- In geometry, the radians of the angle from East to the line vector going from $(x_1, y_1)$ and $(x_2, y_2)$ is $\arctan(\frac{y_2-y_1}{x_2-x_1})$ if $x_2 - x_1$ is positive and $\arctan(\frac{y_2-y_1}{x_2-x_1}) + \pi$ if $x_2 - x_1$ is negative.
- In geometry, the distance from $(x_1, y_1)$ to $(x_2, y_2)$ is $\sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$.
- In ML, the library has `Math.atan`, `Math.sqrt`, and `Math.pi`, which are what you think.
- Use another (possibly local) function. Given the next position, and the current position, compute the new angle and the distance. Given the current angle, turn left the difference between the new and current angle.
- It is fine to turn negative angles.

(d) **Extra Credit** Define a function `remove_dizziness` that takes a list of moves and evaluates to a list of moves. Both lists must cause one to visit the same positions in the same order. But the result list must have these properties:
- Every `Forward` move has a distance strictly greater than 0.
- Every `Turn` move has an angle strictly greater than 0 and less than or equal to $\pi$.
- It may not have more `Turn` moves than the original list has *total* moves.
- It may not have more `Home` moves than the original list.

Hints:
- You may find `Real.abs`, `Real.rem`, and `Real.minPos` useful.
- Because floating-point is subject to rounding errors, you should never compare two floating-point numbers for equality. To test if a value of type `real` "is" `0.0`, you need to see if its absolute value is less than the minimum positive floating-point number (`Real.minPos`).

**Type Summary:** Evaluating a correct homework solution should include these bindings (but it is fine for your definition of `chip_summary` to appear where `chip_summary` appears below because the two types are synonyms):

```
val chip_list_to_summary = fn : chip list -> chip_summary
val summary_to_chip_list = fn : chip_summary -> chip list
val chip_list_value_1 = fn : chip list -> int
val chip_list_value_2 = fn : chip list -> int
val advance = fn : (real * real) * real * real -> real * real
val end_point = fn : move list -> real * real
val visit = fn : (real * real) list -> move list
```

Of course, generating these bindings does not guarantee that your solutions are correct: *Test your functions.*

**Turn-in Instructions**

- Put all your solutions in one file, `lastname_hw2.sml`, where `lastname` is replaced with your last name.
- Line 1 of your `.sml` file should include an ML comment with your name and the phrase `homework 2`.
- Email your solution to `martine@cs.washington.edu`.
- The subject of your email should be *exactly* `[cse341-hw2]`.
- Your `.sml` file should be an *attachment*.