# Homework 1: Tactics for Programming in New Languages

See Webcourses and the syllabus for due dates.
In this homework you will note techniques that you found were helpful for learning new programming languages [Concepts] [UseModels] [MapToLanguages].

## General Directions

Answers to English questions should be in your own words; don't just quote from articles or books.

## What to turn in

For problems that require an English answer, upload your answer to Webcourses for the assignment corresponding to that problem.

## Problems

1. (10 points) [Concepts] [UseModels] [MapToLanguages]. Participate in the discussion on Webcourses titled "HW1, problem 1: Tactics for Programming in Haskell"

   The *tactics* we are concerned with in this problem are programming techniques that are useful for accomplishing small-scale goals in a program. For example, in Java we might have a tactic like the one shown in Figure 1.

### Sequential Search Tactic

(i) **Goal:** To find an element of an indexed data structure (such as an array).
(ii) **Implementation:** Use a for-loop to search through the data structure. For example, in Java this might look as follows.

```
found = false;
for (int i = 0; i < a.length; i++) {
  if (a[i] == sought) {
    found = true;
  }
}
```

Figure 1: An example tactic.

This kind of small scale coding pattern is what we are looking for this problem.

This problem is asking you to reflect on what you are learning in class about tactics for Haskell: what they are and how they are different than what you learned in other languages, such as C and Java. The reason for this is that inappropriate application of tactics, by using tactical plans that are wrong for the new language you are learning, is a major source of difficulty in using a new language [SW90].

Note that a tactic is not a programming language feature (such as list comprehension, pattern matching, lambda functions, etc.). These features can certainly form the basis of programming tactics, but the tactics reside in how the features are *used*, not in the features themselves.

What you are to do is to post to the discussion thread:

1. A tactic, which describes both: (i) the tactic's goal and (ii) how the tactic is carried out (implemented). Your post should include both of these elements and also (iii) give a concrete example to illustrate the tactic.
2. A brief but clear explanation of either (i) why the tactic is important for the new language or (ii) how the tactic is different from what you might use to accomplish the same goal in a language like C or Java that you are familiar with.

(You can also post replies to other, but that is not necessary, and replies about other people's tactic posts will not be graded.)

These posts will be graded based on the quality and originality of your tactic and the reason you give for why it is important or different. We will take points off for: incorrect examples, examples that are not concrete or not very good, a lack of comparison with other languages, and lack of detail in comparison with other languages.

2. (20 points) [MapToLanguages]. Make a table of translations that gives examples of how to translate from Haskell into either Java, C, or C++ (say which). Figure 2 shows how this might look (when the translation target is Java), although your table should have at least 10 entries that are different than the ones shown in this example (and can translate to a different language). Your table must include at least

Translation Table

| Haskell | Java |
|---|---|
| a ‘rem‘ b | a % b |
| if b then e1 else e2 | (b ? e1 : e2) |
| ... | ... |

Figure 2: Example of translations from Haskell to Java

10 different kinds of expressions, statements, or declarations in Haskell, along with their translations into Java or C or C++.

You can include notes limiting the applicability of the translation or setting conditions (such as that new variables chosen must be fresh).

We will give you two points for each correct translation. We will take points off if you confuse statements with expressions or make syntax errors.

3. (10 points) [Concepts] [UseModels] [MapToLanguages]. Participate in the discussion on Webcourses titled “HW1, problem 3: Tactics for Programming in Erlang”

As in the similar problem about Haskell tactics above, the *tactics* we are concerned with in this problem are programming techniques that are useful for accomplishing small-scale goals in a program (e.g., see Figure 1 on the preceding page).

This problem is asking you to reflect on what you are learning in class about tactics for Erlang: what they are and how they are different than what you learned in other languages such as C and Java. The reason for this is that inappropriate application of tactics, by using tactical plans that are wrong for the new language you are learning, is a major source of difficulty in using a new language [SW90].

What you are to do is to post to the discussion thread “HW1, problem 3: Tactics for Programming in Erlang” is:

1. A tactic, which describes both: (i) the tactic's goal and (ii) how the tactic is carried out (implemented). Your post should include both of these elements and also (iii) give a concrete example to illustrate the tactic.
2. A brief but clear explanation of either (i) why the tactic is important for the new language or (ii) how the tactic is different from what you might use to accomplish the same goal in a language like C or Java that you are familiar with.

(You can also post replies to other, but that is not necessary, and replies about other people's tactic posts will not be graded.)

These posts will be graded based on the quality and originality of your tactic and the reason you give for why it is important or different. We will take points off for: incorrect examples, examples that are not concrete or not very good, a lack of comparison with other languages, and lack of detail in comparison with other languages.

4. (20 points) [MapToLanguages]. Make a table of translations that gives examples of how to translate from Erlang into either C, Java, or C++ (say which). Figure 3 shows how this might look, when the translation target is C, although your table should have at least 10 entries that are different than the ones shown in this example and might translate into Java or C++ or C. Your table must include at least 10 different kinds of expressions, statements, or declarations in Erlang, along with their translations into Java or C or C++.

   Translation Table

   | Erlang | C |
   |---|---|
   | a rem b | a % b |
   | if b -> e1; true -> e2 end | (b ? e1 : e2) |
   | ... | ... |

   Figure 3: Example of translations from Erlang to C

   You can include notes limiting the applicability of the translation or setting conditions (such as that new variables chosen must be fresh).

   We will give you two points for each correct translation. We will take points off if you confuse statements with expressions or make syntax errors.

## Points

This homework's total points: 60.

## References

[SW90] Jean Scholtz and Susan Wiedenbeck. Learning second and subsequent programming languages: A problem of transfer. *International Journal of Human-Computer Interaction*, 2(1):51–72, 1990.