Fundamentals of Computer Science I (CS151.02 2007S)

Conditional Expressions

Summary: In this lab, you will have the opportunity to explore Scheme's two primary conditional control operations, `if` and `cond`.

Contents:

Exercises

Exercise 0: Preparation

You may find it helpful to rescan the readings on Boolean values and numbers.

After making sure that you're prepared, start DrScheme.

Exercise 1: Who Won?

Define and test a Scheme procedure, `(report-victory` *`score`*`)`, that takes one argument, a real number, and returns the symbol `won` if *score* is positive, the symbol `lost` if it is negative, and the string `tied` if it is zero.

a. Use `if` for all the tests within your procedure.

b. Use `cond` for all the tests within your procedure.

Exercise 2: Categorizing Values

Write a procedure, `(type-of *val*)`, that returns

- the symbol `boolean`, if *val* is a boolean;
- the symbol `character`, if *val* is a character;
- the symbol `number`, if *val* is a number;
- the symbol `procedure`, if *val* is a procedure;
- the symbol `string`, if *val* is a string;
- the symbol `symbol`, if *val* is a symbol; or
- the symbol `miscellaneous`, if *val* is anything else.

Exercise 3: Categorizing Lists

Define and test a Scheme procedure `list-type` that takes one argument, a list, and returns the symbol `empty` if the argument is the empty list and the symbol `non-empty` otherwise.

Exercise 4: The Sphinx's Riddle

As you may know, one of the famous riddles of the Sphinx goes something like the following:

> What is it that walks upon four legs, then two legs, then three legs?

The answer is, of course, “humans”.

Write a Scheme predicate, `legs`, that, given someone's age, tells how many legs they walk upon. (You get to choose reasonable ages for the three phases of life.)

Exercise 5: Who Won? Revisited

You may recall a previous question that asked for the following: Define and test a Scheme procedure `(report-victory *score*)` that takes one argument, a real number, and returns the symbol `won` if *score* is positive, the symbol `lost` if it is negative, and the string `tied` if it is zero.

In that problem, you wrote one version that used `if` and one version that used `cond`. However, it is also possible to write `report-victory` using only `and`, `or`, and `not`. Try doing so.

If you are confused, you might want to look at the notes on this problem.

For Those With Extra Time

Extra 1: Validating Dates

Write a procedure, `(valid-date? month day)`, that returns `#t` if the numbers *month* and *day* describe a valid month and day and false otherwise. For example,

```
> (valid-date? 1 30)
#t
> (valid-date? 2 30)
#f
> (valid-date? 9 30)
#t
> (valid-date? 9 31)
#f
> (valid-date? 9 -1)
#f
> (valid-date? 9 250)
#f
```

a. Write this procedure using `if` as the primary control structure.

b. Write this procedure using `cond` as the primary control structure.

c. Write this procedure without `if` and `cond` (that is, using `and`, `or`, and, possibly, `not`).

Exercise 2: Classifying Numbers

Write a procedure, `(classify-number num)`, that creates a string that classifies *num* according to its exactness and its least general type (selected from `integer`, `rational`, and `complex`).

```
> (classify-number 25)
"exact integer"
> (classify-number -3.5)
"inexact rational"
> (classify-number 3+4i)
"exact complex">
```

You can assume that *num* is a number.

Exercise 3: Classifying Numbers, Revisited

Write a procedure, `(classify-number num)`, that returns a string that describes *num* subject to the following restrictions.

- Describe zero as `"zero"`;
- Just as in the previous problem, return the return the exactness of the value (that is, results should begin with `exact` or `inexact`:
- For non-complex values, include the sign (positive or negative);
- For exact rational/real values, use the type “rational”; and
- For inexact rational/real values, use the type “real”.

For example,

```
> **(classify-number 23)**
"exact positive integer"
> (classify-number #i23)
"inexact positive integer"
> (classify-number -2.3)
"inexact negative real"
> (classify-number 24/7)
"exact positive rational"
```

Notes

Notes on Exercise 5

To write the procedure `report-victory` without `if` and `cond`, we need to take advantage of the way `and` and `or` behave. Recall that `and` either returns false (if it encounters a false) or the last value (if it fails to encounter a false value) and that `or` returns the first non-false value, if one is there.

Hence, we'll use an `and` for each test and group them together within an `or`.

The first test looks something like this: `(and (> score 0) 'won)`. Note that this test returns `#f` if *score* is less than or equal to 0 and the symbol `won` otherwise.

We can then put that in an `or` expression. Here's a start.

```
(or (and (> score 0) 'won)
    'something-else)
```

Now, if *score* is greater than 0, we get `won` and otherwise we get `something-else`.

The remainder is up to the reader to figure out.

Return to the problem.