

CIS-280
Lab 7: Data Directed Programming
Monday, April 11, 2005

File `~carberry/Lab-7.scm` contains the procedures `Get-Type` and `Get-Rep` discussed in class. It also contains some procedures that will implement the `get` and `put` operations that we discussed in class. You will want to copy this file into your own directory and call it `Lab-7`. In the following, you are going to add to the code in `Lab-7`.

1. Load the code in `Lab-7` into Scheme.
2. Define a procedure (`smallest-unordered set`) that takes as argument a set that is an unordered list and returns the smallest element in the set. (You may write more than one procedure.) Thus (`smallest-unordered '(5 3 7 2 6)`) returns 2.
3. Define a procedure (`largest-unordered set`) that takes as argument a set that is an unordered list and returns the largest element in the set. (You may write more than one procedure.) Thus (`largest-unordered '(5 3 7 2 6)`) returns 7.
4. Define a procedure (`smallest-ordered set`) that takes as argument a set that is an ordered list and returns the smallest element in the set. Thus (`smallest-ordered '(3 7 12 14)`) returns 3.
5. Define a procedure (`largest-ordered set`) that takes as argument a set that is an ordered list and returns the largest element in the set. Thus (`largest-ordered '(3 7 12 14)`) returns 14.
6. Define a procedure `construct-table` that puts the procedures `smallest-ordered`, `smallest-unordered`, `largest-ordered`, and `largest-unordered` into a table using the `put` function. Note that procedure `construct-table` has no arguments. The columns of the table should be the type (either `unordered` or `ordered`) and the rows of the table should be the operation (either `smallest` or `largest`)
7. Now execute your procedure `construct-table`. This sets up your table of procedures as we discussed in class. (Whenever you press the `Execute` button to execute the definitions in your definition window, you will need to again execute `construct-table` in the execution window. Otherwise, when you try to retrieve a procedure from the table, it will return the empty list since the table has been erased.)
8. Now for practice, execute each of the following and make sure you understand what you got and why:

```
(get 'ordered 'smallest)
```

```
( (get 'ordered 'smallest) '(4 6 7 10))
```

```
( (get 'ordered 'largest) '(4 6 7 10))
```

9. Now write a procedure (`small-small set1 set2`) that takes as arguments two sets `set1` and `set2` and returns the sum of their smallest elements. `set1` and `set2` will have manifest

type, but their types may be different. Procedure `small-small` should check the type of `set1`, extract the appropriate procedure from the table, execute the extracted procedure on `set1` after its type is removed, check the type of `set2`, extract the appropriate procedure from the table, execute the extracted procedure on `set2` after its type is removed, and then sum the two results.

10. Now execute the following:

```
(define seta (Add-Type 'ordered '(5 7 18 24)))
(define setb (Add-Type 'unordered '(35 22 4 39 3 67)))
(define setc (Add-Type 'ordered '(7 15 30)))
(define setd (Add-Type 'unordered '(45 4 67 34)))
```

11. Execute `(small-small seta setb)` — if you have done this correctly, it should return 8
12. Execute `(small-small setb setd)` — if you have done this correctly, it should return 7.
13. Execute `(small-small setb setc)` — if you have done this correctly, it should return 10.
14. Submit your code for Lab-7 on the electronic submission system and see if you've done everything correctly.