Travers Parsons-Grayson
George Witteman
Yina Wang
Spring 2018

Auto-Grader Instruction Manual

Appendix I: List of Files

The auto-grader consists of several files. The purpose of each is outlined below:

I **File: "format.txt"** The "format.txt" file contains various functions allowing for organized and aligned printing of students' results.

II **File: "structs-and-helper-funcs.txt"** This file contains various helper functions used throughout our main functions. The structs used throughout the auto-grader are also defined and explained within this file.

III **File: "grader.txt"** Contained in this file are functions for printing students' results and for checking the correctness of the code. The main function **grade-asmt** is to be run from this file.

IV **File: "test-template.txt"** A template file for the test file the instructor must define.

Appendix II: Instructor Use

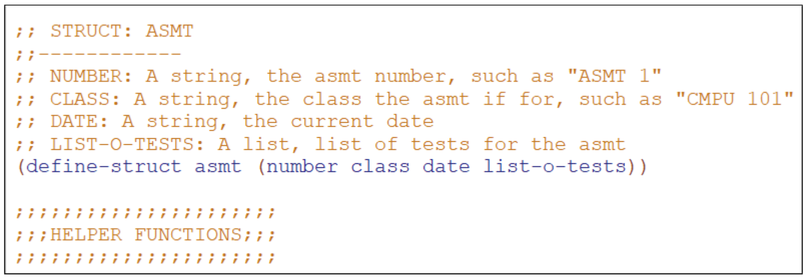
I **Creating the Test File**

i First open the test template, "test-template.txt". At the top of the file (if it is not already there), type the following code: **(load "structs-and-helper-funcs.txt")**.

ii Next the instructor must define an assignment structure. See Figure 1 below for reference.

Creating the first three fields of the ASMT struct is self-explanatory; however creating a list of tests requires a little more work. See instructions below on creating a TEST struct. Reference Figure 2 below for help.

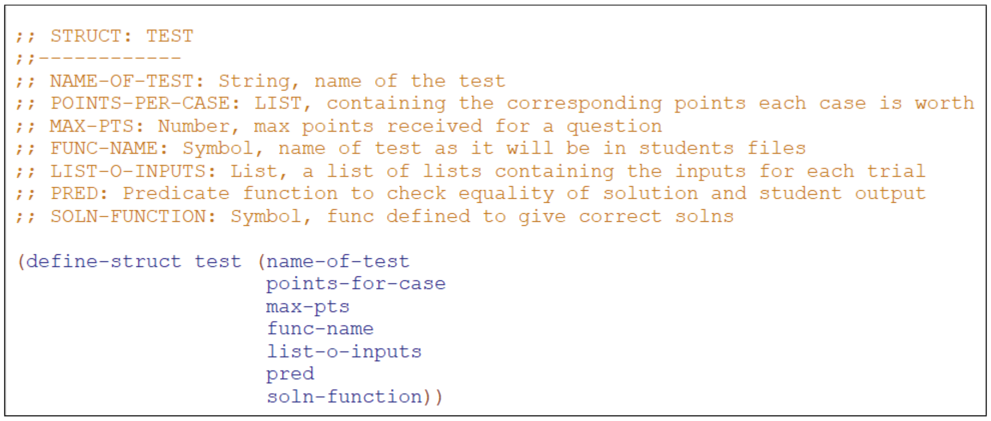
Note: The points-for-case and list-o-inputs parameters should be lists of equal lengths. Points-for-case is a list of points that the instructor can designate per input test case. i.e. points-for-case[i] corresponds to the number of points that a correct answer for list-o-inputs[i] is worth.



```
;; STRUCT: ASMT
;;------------
;; NUMBER: A string, the asmt number, such as "ASMT 1"
;; CLASS: A string, the class the asmt if for, such as "CMPU 101"
;; DATE: A string, the current date
;; LIST-O-TESTS: A list, list of tests for the asmt
(define-struct asmt (number class date list-o-tests))

;;;;;;;;;;;;;;;;;;;;;;
;;;HELPER FUNCTIONS;;;
;;;;;;;;;;;;;;;;;;;;;;
```

Figure 1: ASMT Structure

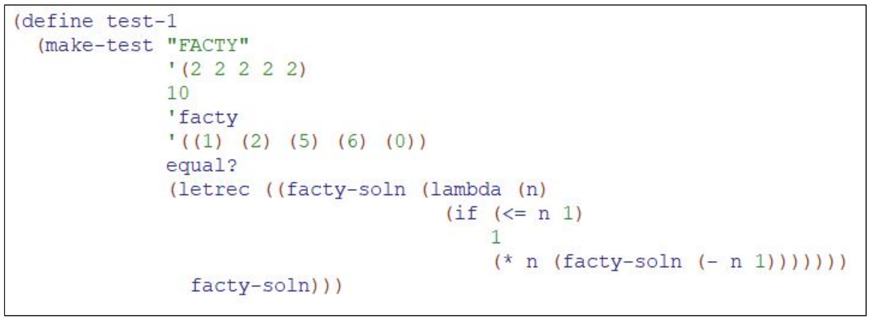


```
;; STRUCT: TEST
;;------------
;; NAME-OF-TEST: String, name of the test
;; POINTS-PER-CASE: LIST, containing the corresponding points each case is worth
;; MAX-PTS: Number, max points received for a question
;; FUNC-NAME: Symbol, name of test as it will be in students files
;; LIST-O-INPUTS: List, a list of lists containing the inputs for each trial
;; PRED: Predicate function to check equality of solution and student output
;; SOLN-FUNCTION: Symbol, func defined to give correct solns

(define-struct test (name-of-test
                     points-for-case
                     max-pts
                     func-name
                     list-o-inputs
                     pred
                     soln-function))
```

Figure 2: Test Structure

iii In the **"test-template.txt"** file, the instructor should begin by loading **"structs-and-helper-funcs.txt"**. Then the instructor can define the necessary TEST structs needed for the assignment by following the example provided in Figure 3.

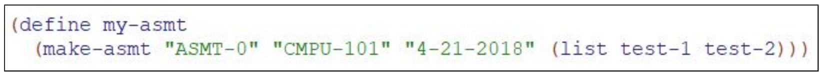


```
(define test-1
  (make-test "FACTY"
             '(2 2 2 2 2)
             10
             'facty
             '((1) (2) (5) (6) (0))
             equal?
             (letrec ((facty-soln (lambda (n)
                                    (if (<= n 1)
                                        1
                                        (* n (facty-soln (- n 1)))))))
               facty-soln)))
```

Figure 3: Defining TESTs

Note: Consider the function (lambda (x y) (* x y)). The list of inputs would look something like this **'((1 2) (0 0) (8 4))**. The inputs for a function are always within their on list. For example for some function (lambda (listy) ....), the input list with be a list of lists of a list. That is, the input list will look something like this **'(((1 2)) ((0 0)) ((8 4)))**

The instructor must define a new TEST struct for each separate function. After defining all the TEST structs, the instructor should use the AMST struct to store the list of tests along with the assignment name, date, and class. See code below.



```
(define my-asmt
  (make-asmt "ASMT-0" "CMPU-101" "4-21-2018" (list test-1 test-2)))
```

Figure 4: Putting TEST structs in ASMT struct

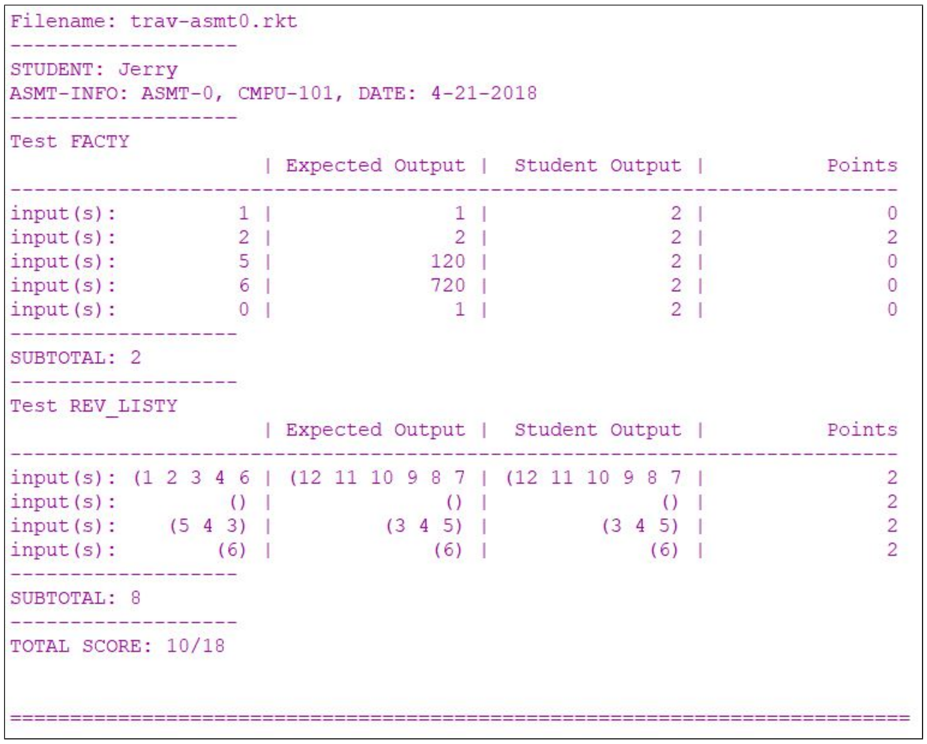
II **Using the Grader**

i Open or run**"grader.txt"** file. In the **Interactions** window, type the following: **(grade-asmt "asmtX" "test-template.txt")**.

The function will fetch all the files with "amstX" in the name to run against the instructor's solution function from "test-template.txt".

ii The assignment information is printed out along with the inputs, expected inputs, students' inputs, and points assigned. After each function are graded, a SUBTOTAL will be printed. After all the functions of one student's assignment are graded, a TOTAL will be printed as total points received out of maximum total points. See Figure 5 below for reference.

iii Each student must define a global variable **student-name** in order for their name to be printed out at the top of their assignment. If you wish to change the name of the global variable for the student name, simply go into in the **grader.txt** file and change all instances of **student-name** to the desired name for your global variable.



```
Filename: trav-asmt0.rkt
-------------------
STUDENT: Jerry
ASMT-INFO: ASMT-0, CMPU-101, DATE: 4-21-2018
-------------------
Test FACTY
                     | Expected Output |  Student Output |          Points
---------------------------------------------------------------------------
input(s):          1 |               1 |               2 |               0
input(s):          2 |               2 |               2 |               2
input(s):          5 |             120 |               2 |               0
input(s):          6 |             720 |               2 |               0
input(s):          0 |               1 |               2 |               0
-------------------
SUBTOTAL: 2
-------------------
Test REV_LISTY
                     | Expected Output |  Student Output |          Points
---------------------------------------------------------------------------
input(s): (1 2 3 4 6 | (12 11 10 9 8 7 | (12 11 10 9 8 7 |               2
input(s):         () |              () |              () |               2
input(s):    (5 4 3) |         (3 4 5) |         (3 4 5) |               2
input(s):        (6) |             (6) |             (6) |               2
-------------------
SUBTOTAL: 8
-------------------
TOTAL SCORE: 10/18


============================================================================
```

Figure 5: Auto-grader Results