COMP1811 – Scheme Project Report

## SOFTWARE DESIGN

*Briefly describe the design of your coursework software and the strategy you took for solving it. – e.g. did you choose either recursion or high-order programming and why…*

*Complete individually. Describe the design of the parts you implemented.*



PARTNER A: Nobel Ahmed

F3.i. First defined event and then formatted this implementing “cons” on it. And it’s formatted such id and params should be in pair and their pair will make another pair with time.

F3. ii. First defined event and then used pair? to check the value is paired or not. integer? Predicate were used to check the first element which was time is integer. car was used to find the first element of the value. member function was called to find an element if an element is presented in the list. Thus, we again use pair and integer function.

F3.iii. I used car function on the value that will be written in the REPL to find the first element which is even-time.

F3iv. To find the type of the event I used cadr on the value.

F3.v. If we use cddr on the value, it will find the pair of the user id and lane id.

F3.vi. car on the cddr of the value will give use the customer id.

F3.vii. cdr on cddr of the value will give us the lane id

F4.i. I used high-order programming on this task. Fist I defined the sim-add-event which will give us a new sorted list from sim-event-queue and ev by using append. We used lambda function to sort the specific number only. Here a and b is compared using if and they were borrowed from event-time. If they are equal, a and b will be compared again but this time they will be borrowed from event-user. If they are not equal, they will be sorted and will not go to second condition. Here since, a and b were borrowed from event-time and event-use and they are defined already in another file so high-order- programming were implemented.



PARTNER B: MD Mahbub Anam Tanim

1st function, sim-secs2hour takes an input in seconds and converts it into hours, minutes, and seconds. It calculates all the thing in a sequential order, calculating total minutes, hours, remaining minutes, and remaining seconds. Then returns a list containing the hours, minutes, and seconds.



F1.

The lane? function makes sure the format of a lane by using different conditions. Then lane-id, lane- user, and lane-queue give the access to ID, user, and queue details respectively. The function lane- bussy? Checks whether the lane is busy or not by using if condition, which helps during the whole simulation. Calculating queue lengths, lane-length helps the queue management and it’s analysis. These functions use car, cdr, cadr, cddr and other necessary procedures to fetch the required data from lane enabling efficient manipulation and examination of lane properties.



F2.

For 1st function, ‘lane-length’, high order programming and recursion were used to solve the problem. First, it checks whether the input is null. Then calls ‘lane?’ from “sim-lane.rkt” Function to check if the 1st lane of the input is a valid lane. After that rest of the lanes are sent back to the start of the function to get checked through again recursively. If all those lanes pass the ‘lane?’ Check test than returns #t.



F5.

As for ‘less-crowded’ function, sort function and high order programming were used among the other. Input passes through two arguments of lambda which than gets compared by calling ‘lane-length’ function from “sim-lane.rkt” and sorted in an ascending order later to print the least crowded lane.

## CODE LISTING

*Provide a complete listing of the entire Scheme code you developed. Make sure your code is well commented, specially stressing informally the contracts for parameters on every symbol you may define. The code listed here must match that uploaded to Moodle. Please copy and paste the actual code – no screenshots please! Make it easy for the tutor to read. Marks WILL BE DEDUCTED if screenshots are included. Add explanatory narrative if necessary – though your in-code comments should be clear enough.*

*Complete as a pair. Highlight which features each partner implemented.*

* 1. F1

i. (define (sim-secs2hour seconds) ; converts seconds into hours, minutes and seconds

;; Calculates total minutes

(let\* ((tot-mins (quotient seconds 60)) ;sequentially creates bindings

;; Calculates hours

(hrs (quotient tot-mins 60))

;; Calculates remaining minutes

(mins (remainder tot-mins 60))

;; Calculates remaining seconds (secs (remainder seconds 60)))

;; Returns a list that contains hours, minutes, and seconds (list hrs mins secs)))

* 1. F2

i. (define (lane-length ln)

;;checks the length of the value of last item of 2nd pair(cdr of cdr) of ln (length(cddr ln)))

Ii. (define (lane? val) ; checks if the input is valid lane

(and (pair? val) ;checkes whether the input is pair and if it meets other conditions

;;checks if the 1st value of input is number (number? (car val))

;;checks if the rest is a pair or not (pair? (cdr val))

;;either the 1st value of 2nd pair is a number or the 2nd pair is null (or (number? (car (cdr val))) (null? (cadr val)))

;;or the last vale of 2nd pair is a list or null (or (list? (cddr val)) (null? (cddr val)))))

1. (define (lane-bussy? val); checks if the till is busy or not (if (null? (cadr val))

#f ; returns false is the till is not busy #t)) ; true if the till is not null

1. (define (lane-id lane)

(car lane)) ;returns the ID of a lane

1. (define (lane-user val)

(car (cdr val))) ; returns the 1st value(ID of the customer) of 2nd pair

1. (define (lane-queue val)

(cddr val)) ; returns the last value (list of customers) of the 2nd pair

* 1. F3

1. (define (event time id params)

(cons time(cons id params)); create the format of event(reformats pairs)

)

1. (define (event? val) ;checkes if the val is a an event (returns #t if so) (and (pair? val) ; must be a pair

(integer? (car val)) ; checks if the 1st element (time) is an integer (>= (car val) 0) ; checks if the 1st element is greater of qual to 0 (pair? (cdr val)) ; the 2nd element must be a pair

(member (car (cdr val)) '(1 2 3)) ;the second element must be either 1,2 or 3 (pair? (cdr (cdr val))) ;3rd element must be a pair

(integer? (car (cdr(cdr val)))) ;1st part of 3rd element must be an integer (integer? (cdr (cdr(cdr val)))))) ;2nt part of 3rd element must be an integer

1. (define (event-time ev)

(car ev)); gets the 1st element of ev (time)

1. (define (event-type ev)

(cadr ev)) ; gets the 1st item of 2nd pair (types of event)

1. (define (event-params ev)

(cddr ev)) ;gets the 2nd item of 2nd element as a pair(customer & lane ID)

1. (define (event-user ev)

(car (cddr ev))) ;returns the customer ID engaged in this event (car of cdr of cdr)

1. (define (event-lane ev)

(cdr (cddr ev))); ;returns the lane ID related to the event (cdr of cdr of cdr)

* 1. F4

i. (define (sim-add-event sim-event-queue ev) ;returns a list of events in ascending order based on conditions

;sorts the merged sim-event-queue(current event) and ev(new list) based on time & user (sort (append sim-event-queue (list ev))

(lambda (a b) ;function with arguments for comparison

(if (= (event-time a) (event-time b)); checks if the event times are equal (< (event-user a) (event-user b)) ;if so, sort by event user

(< (event-time a) (event-time b))))));else, sort by event time

* 1. F5

1. (define (lane-list? lst) ; checks if the input is a valid list of lanes (if

(null? lst) ; checkes whether the input is null #t

;;checks if the 1st lane of the list is a valid lane

;;and pass the rest of the lanes to lane-list? for checking sequentially using a recursive call (lane? (first lst)) (lane-list? (rest lst)))))

1. (define (less-crowded sim-lanes) ; gets the least crowded lane from list of lanes

(car (sort sim-lanes ; gets the 1st lane after sorting (sorted in an ascending way, least to most crowded) (lambda (a b) ;takes two arguments for comparison with different with conditions

(< (lane-length a)(lane-length b)) ; sorts in an ascending order

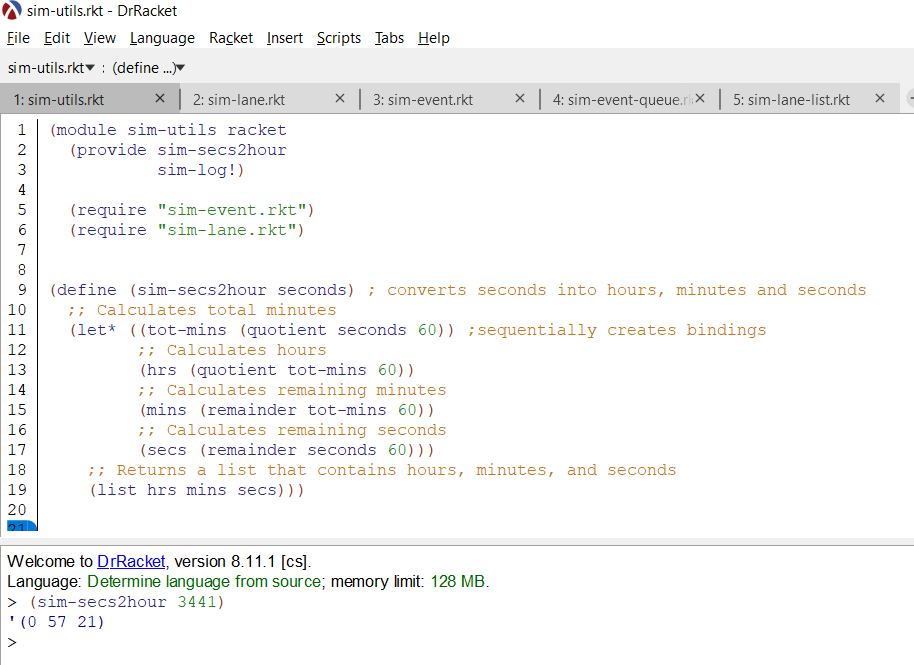
))))

## RESULTS – OUTPUT OBTAINED

*Provide screenshots that demonstrate the results generated by running your code. That is show the output obtained in the REPL when calling your functions. Alternatively, you may simply cut and paste from the REPL.*

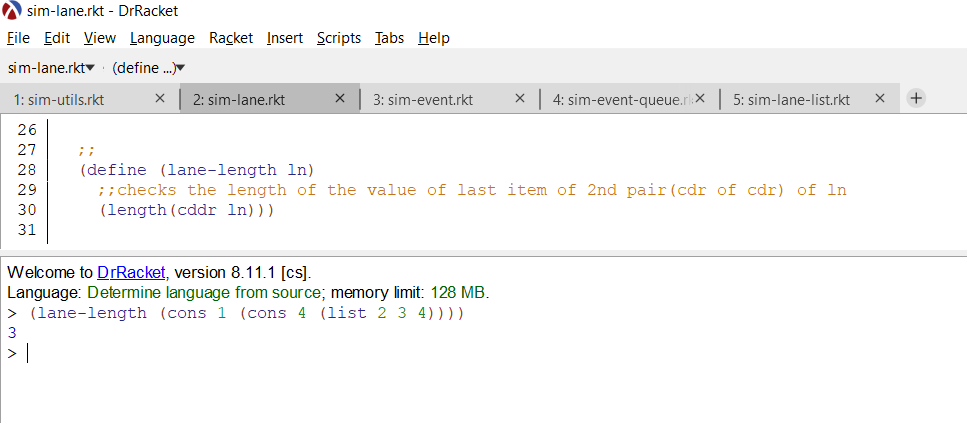
* 1. F1

# sim-secs2hour

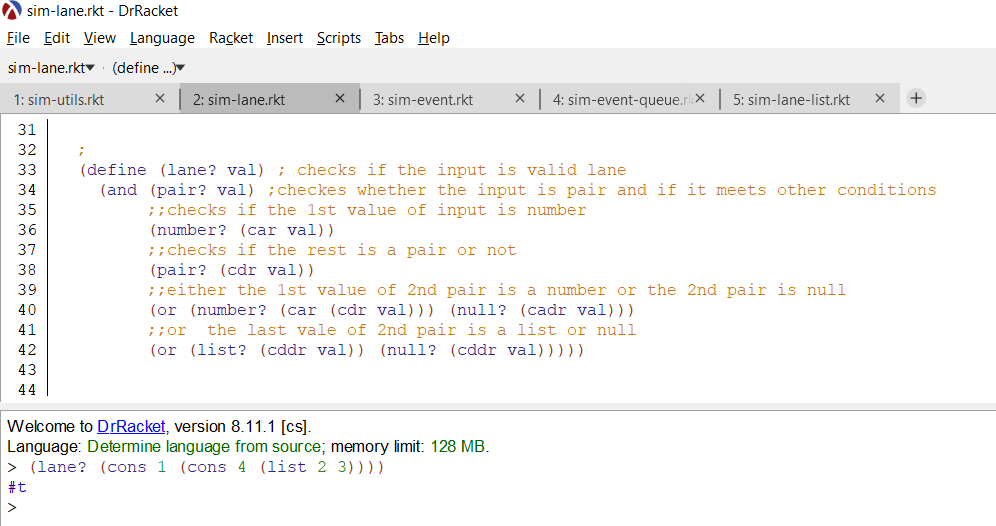


* 1. F2

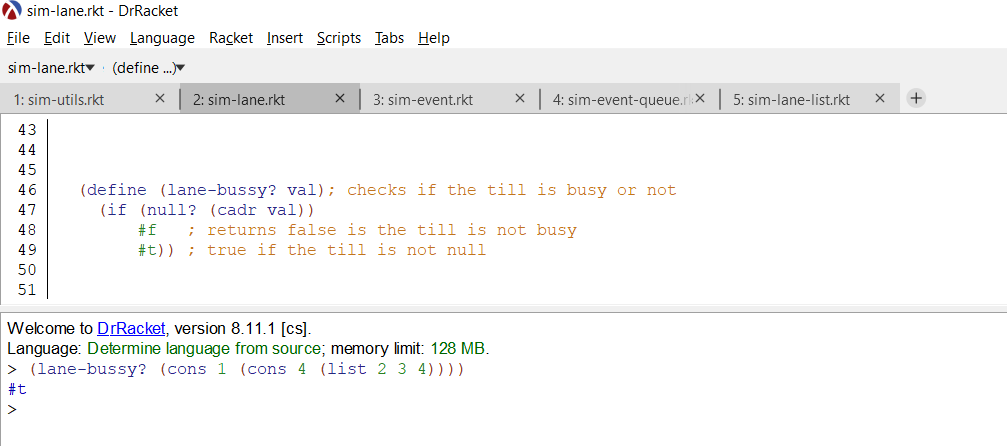
# I. lane-length



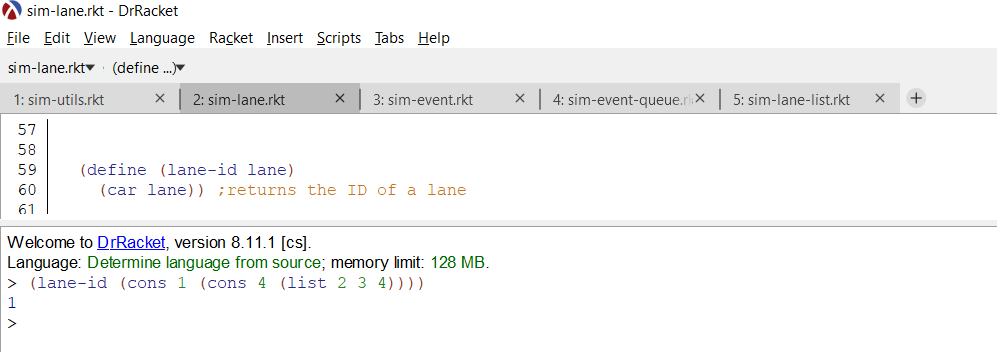
1. lane?



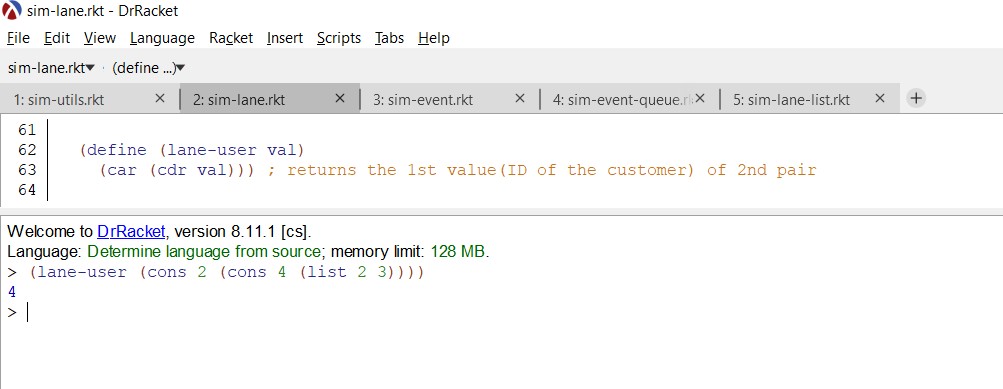
# lane-bussy?



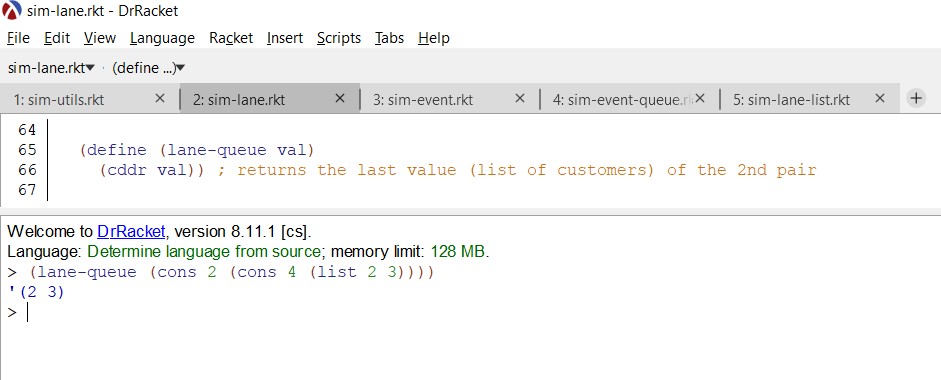
1. lane-id



# lane-user

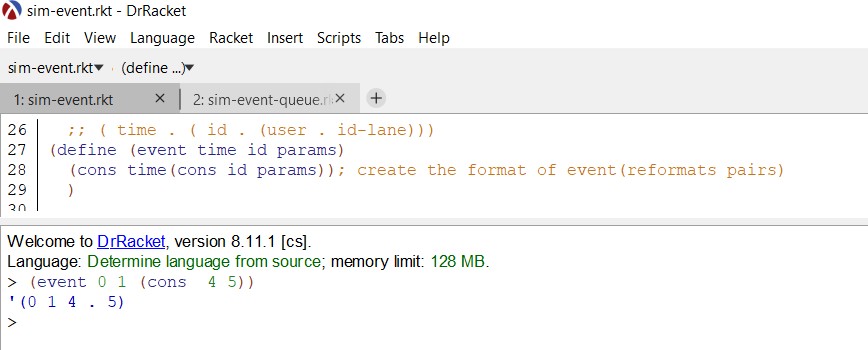


1. lane-queue

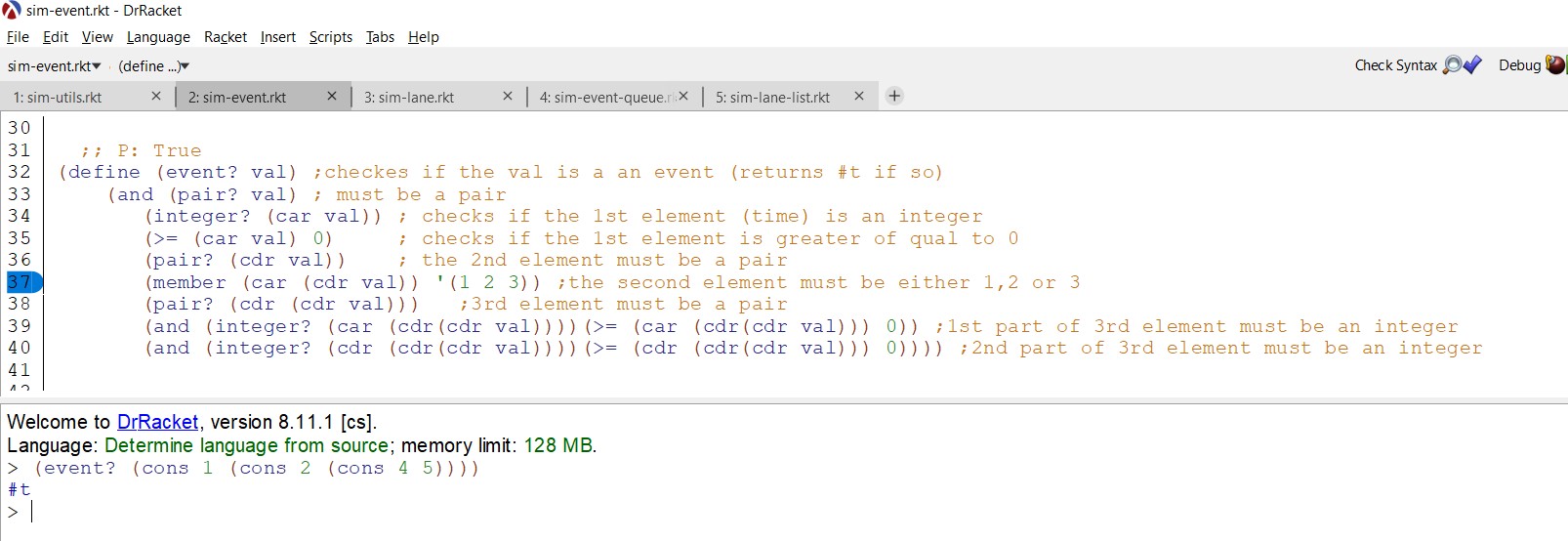


* 1. F3

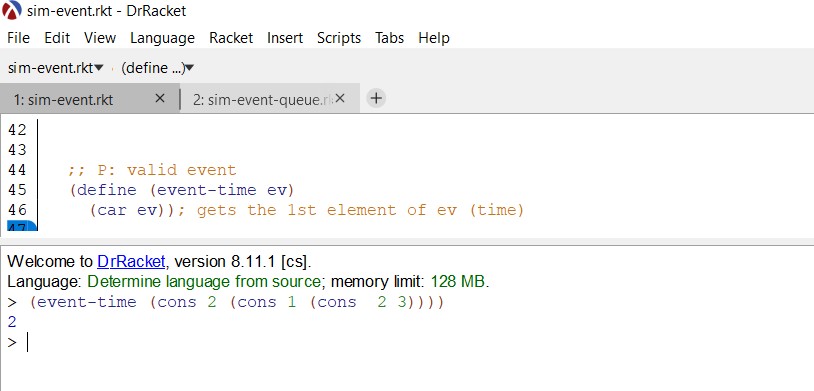
# event



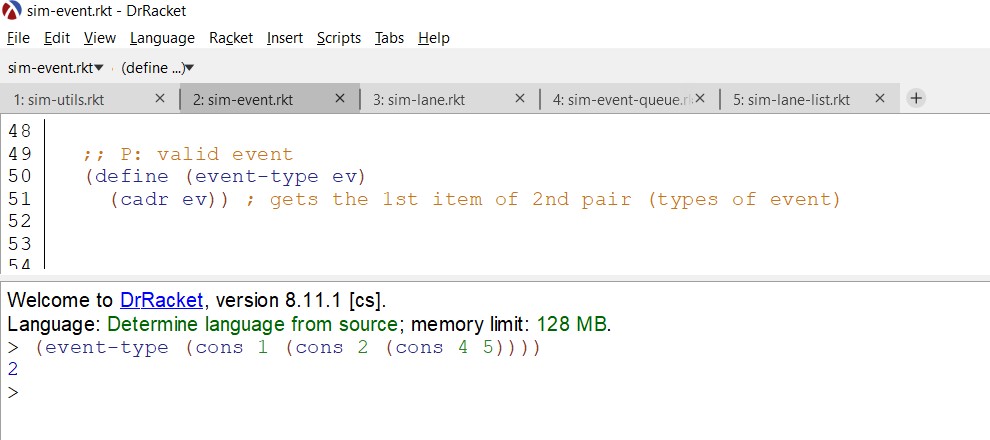
1. event?



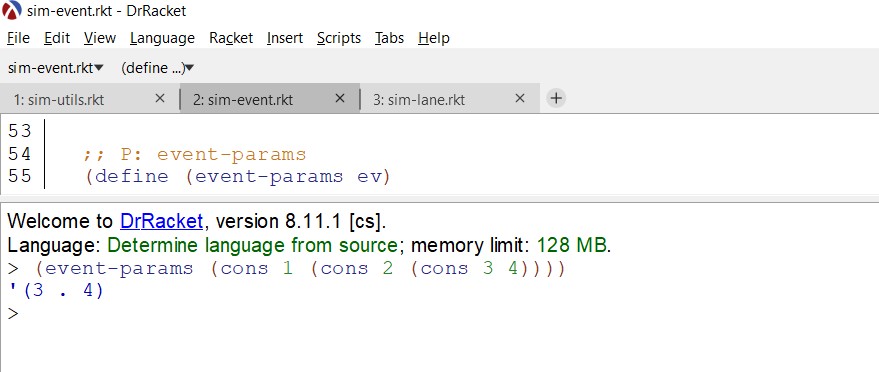
# event-time



1. event-type



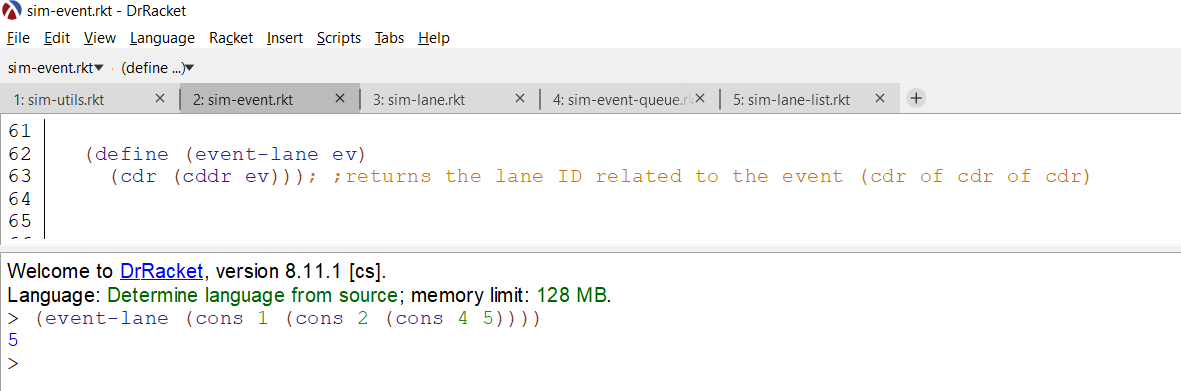
# event-params



1. event-user

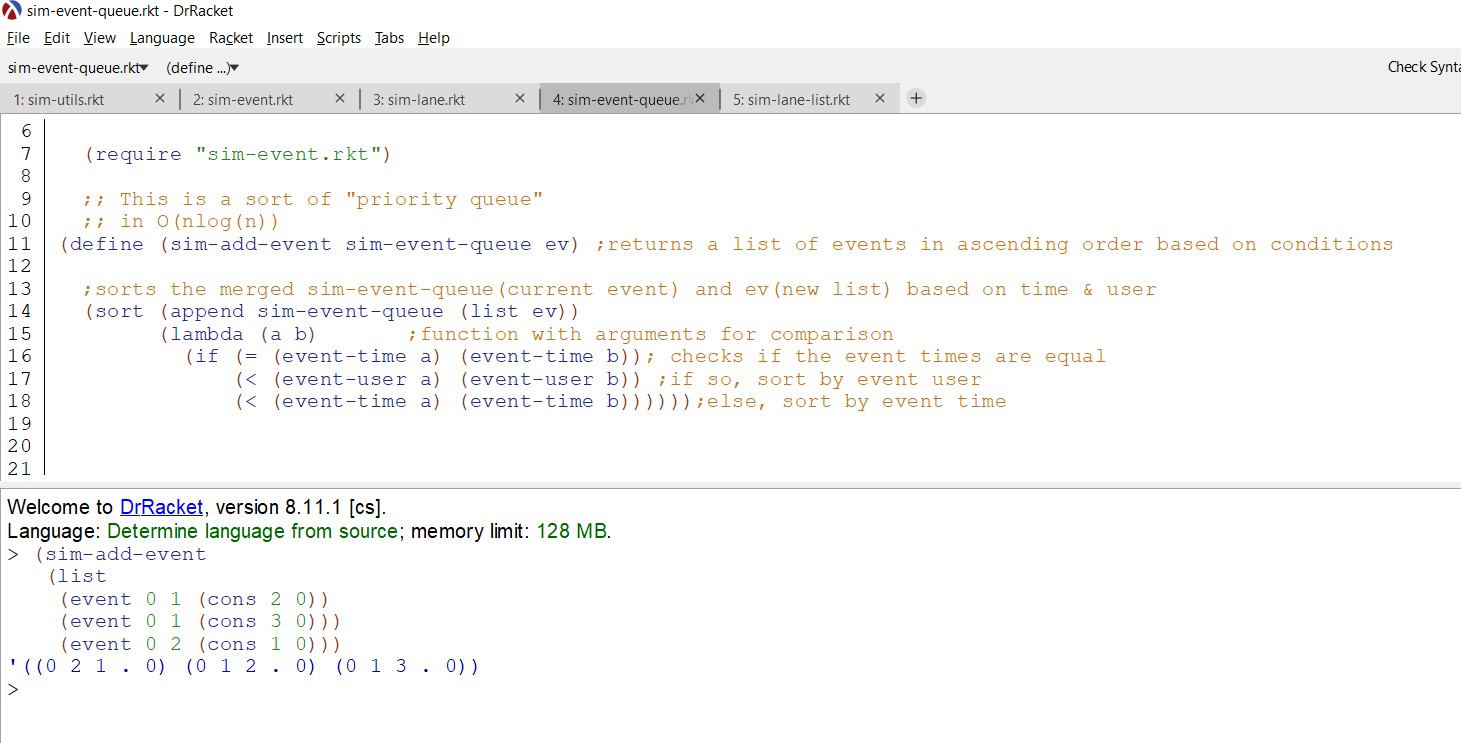


# event-lane



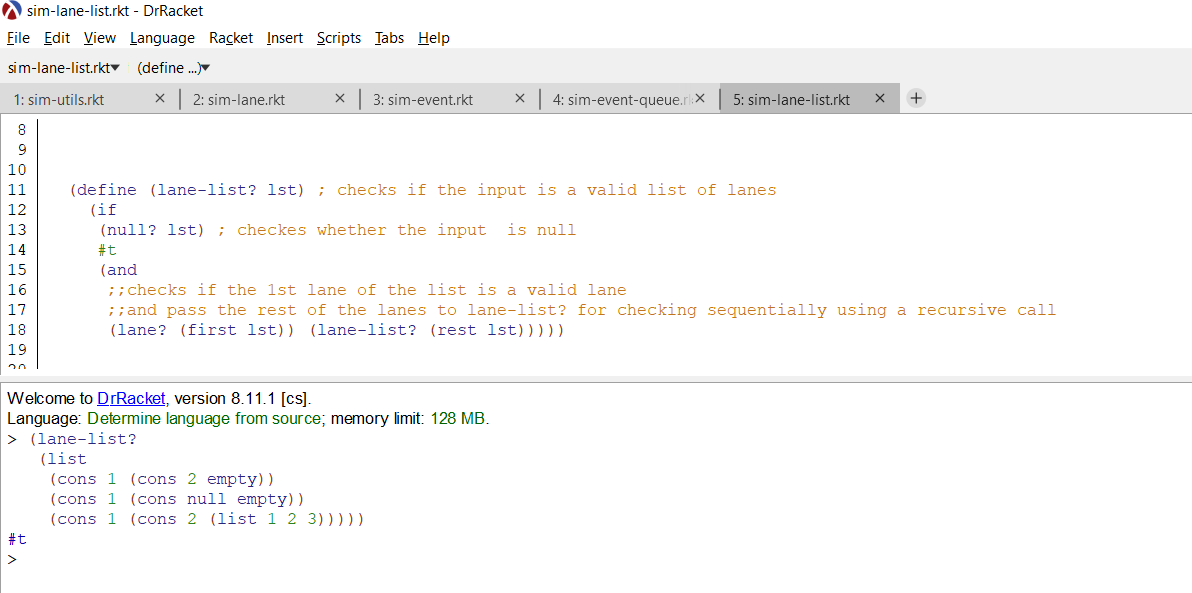
* 1. F4

# i. sim-add-event

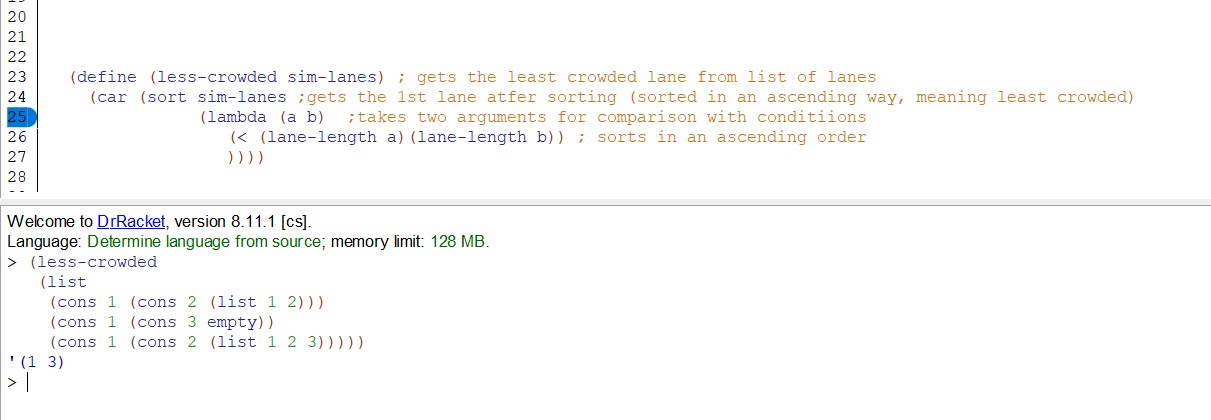


* 1. F5

# lane-list?



1. less-crowded



1. TESTING

*Provide a test plan covering all of your functions and the results of applying the tests.*

For testing of F1, F2 and F5, different test cases were used to check the validation of each function. For instance, in some cases a pair was deliberately not given to check, for other functions symbols were given instead of number, null value, empty lists were used to check whether the functions were working properly. Below, screenshots of different test cases for each function have been provided below.

Same criteria were followed for F3, F4

Below, the functions, their test cases and their output has been put in a table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function name | Input | Output | Justification | Analysis |
| F1.i sim- secs2hour | 4991 | '(1 23 11) | Works properly. Gets proper hour, minutes, and seconds count.  Quotient, remainder working as expected | expected |
| F1.i sim- secs2hour | 0 | ‘(0 0 0) | As expected nothing is divisible so it returns 0 for each part | expected |
| F1.i sim- secs2hour | 60 | ‘(0 1 0) | Divides and get a minute | expected |
| F2.i.lane-length | (lane-length (cons 1 (cons 3 (list 2 3 4)))) | 3 | Returns the length of the list which is 3 in this case | expected |
| F2.i.lane-length | (lane-length (cons 1 (cons 3 (list ) ))) | 0 | Returns the length of the list which is 0 in this case because it doesn’t contain any member. | expected |
| F2.ii.lane? | (lane? (cons 1 (cons 3 (list  2 3 4)))) | #t | Meets the pair requirements,  whether it’s a list or not and test each number for validation | expected |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| F2.ii.lane? | (lane? (cons 'a (cons 3 (list 2 3 4))) | #f | Returns false because first item must be a number, where it is a symbol in this case | expected |
| F2.ii.lane? | (lane? (cons 1 (cons 3 ))) | Error: arity mismatch, expected 2,  given 1 | Format of pair doesn’t match. | expected |
| F2.iii.lane-bussy? | (lane-bussy? (cons 1 (cons 3 (list ) ))) | #f | List doesn’t have any item in it | expected |
| F2.iii.lane-bussy? | (lane-bussy? (cons 1 (cons 3 (list 1 2 3) ))) | #t | Checks the till by car of cdr, whether it is null or empty.  Returns true because till is busy. | expected |
| F2.iv.lane-id | (lane-id (cons 1 (cons 3  (list 2 3 4)))) | 1 | Gets the 1st item of pair as a ID of the lane,1 in this case | expected |
| F2.iv.lane-id | (lane-id (cons 3(cons 3 (list 2 3 4)))) | 3 | Gets the 1st item of pair as a ID of the lane, 3 in this case | expected |
| F2.v.lane-user | (lane-user (cons 3 (cons 2  (list 3 4)))) | 2 | Shows the Id of customer by car of cdr, 2 in this case | expected |
| F2.v.lane-user | (lane-user (cons 3 (cons 1  (list 3 4)))) | 1 | Shows the Id of customer by car of cdr, 1 in this case | expected |
| F2.vi.lane-queue | (lane-queue (cons 3 (cons 1 (list 3 4)))) | ‘(3 4) | Returns the cddr value of input showing the customer waiting in the queue | expected |
| F2.vi.lane-queue | (lane-queue (cons 3 (cons 1 (list 1 2 3 4)))) | ‘(1 2 3 4) | Returns the cddr value of input showing the customer waiting in the queue | expected |
| F3.i event | (event 0 1(cons 4 5)) | '(1 4 . 5) | The original result was (0 1 4 . 5). For  the testing I removed | expected |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | “cons time” to check if the code is running or not. So, it gave me the output  (1 4 . 5) meaning the first element was removed from the original result and it is working as expected |  |
| F3.ii event? | (event? (cons 1(cons 3(cons 4 6)))) | #f | When we type this function, the original code gives the result “#t”. For the testing purpose I changed the condition integer to string. As per the modified code condition the first element should be string which is originally integer in the coursework input. So it results “#f” which is expected and the  code is running well. | expected |
| F3.iii. event-time | (event-time (cons 1(cons 3(cons 4 6)))) | 3 | As per the original code the result should be 1 because the first element is defined as time.  However, the modified code returns the result 3 because I putted an extra cdr on the value and then car on it so it is giving the wrong output. | expected |
| F3.iv. event-type | (event-type (cons 1(cons 3(cons 4 6)))) | 4 | The event type is originally associated with the first element of the second pair but in the modified  code it’s associated | expected |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | with the first element of the 3rd pair. And that’s why it gives  the wrong output. |  |
| F3.v. event- params | (event-params (cons 1(cons 3(cons 4 6)))) | 6 | Similarly, the param should give a pair originally but since I associated it with the last element of the 3rd pair by using “cdr(cdr(cd ev))”, it returns the wrong output which is 6. |  |
| F3.vi. event-user | (event-user (cons 1(cons 3(cons 4 6)))) | '(4 . 6) | The original result should be the user id which is first of the 3rd pair but since it is associated with the pair, it shows the  wrong output. | expected |
| F3.vii. event-lane | (event-lane (cons 1(cons 3(cons 4 6)))) | 4 | For changing the cdr into car from the original code it  returns 4 instead of 6 | expected |
| F4.i. sim-add- event | (sim-add-event (list  (event 0 1(cons 2 0))  (event 0 1(cons 3 0)))  (event 0 2(cons 1 0))) | '((0 1 3 . 0) (0 1  2 . 0) (0 2 1 .  0)) | If the time is same for all users, they should be placed in ascending order according to their ID. But in the modified code condition I putted the descending condition and so it’s giving me the wrong result. | expected |
| F4.ii. Sim-add- event | (sim-add-event (list  (event 0 1(cons 2 0))  (event 10 1(cons 3 0)))  (event 5 3(cons 1 0))) | '((10 1 2 . 0) (5  3 1 . 0) (0 1 3 .  0)) | Similarly, if the time is not same the output should be ascending but since the code is changed it is giving descending order which is wrong | expected |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| F5.i. lane-list? | (lane-list? (list  (cons 1 (cons 2 empty))  (cons 1 (cons null empty))  (cons 1 (cons 2 (list 1 2  3))))) | #t | It recursively checks for validity of each lane whether they are in a formatted order and shows output, shows #t in this case | expected |
| F5.i. lane-list? | (lane-list? (list  (cons 1 (cons 2 empty))  (cons 'a (cons 3 empty)) (cons 1 (cons 2 (list 1 2  3))))) | #f | Returns #f because 2nd lane in this lane doesn’t match the formatted order of a lane | expected |
| F5.ii. less- crowded | (less-crowded (list  (cons 1 (cons 2 empty))  (cons 1 (cons null empty))  (cons 1 (cons 2 (list 1 2  3))))) | '(1 ()) | Grabs the least crowded lane by sorting it in an ascending way and shows the 1st item of the sorted list.  As for this example, it grabs '(1 ()) which is the least crowded lane | expected |
| F5.ii. less-crowed | (less-crowded (list  (cons 1 (cons 2 empty))  (cons 1 (cons 2 (list 1  2)))  (cons 1 (cons 2 (list 1 2  3))))) | '(1 2) | Grabs the least crowded lane by sorting it in an ascending way and shows the 1st item of the sorted list.  For this part, it grabs ‘(1 2) lane which is the least crowded list | expected |

The sim-sim.rkt file was ran to check whether the whole simulation is working properly and has been cross checked the simulation output with the video provided by Mr. Rafael. A proof of the simulation working properly has been provided below. The simulation had 3 lanes and 3 customers.

