Midterm Study Guide: 270

The midterm will take place in the classroom (1052) during your section's the normal class time on Monday July 31, 10a-12p. Late midterms are not accepted, and you must be present in class to take the midterm, unless alternate arrangements were made ahead of time.

There are ten questions on the midterm. They go approximately in the order of what in my opinion is easiest to hardest (but obviously that is subjective). The first three are worth 5pts each, the last three are worth 15pts each, and the middle four questions are 10pts each. The last two questions (i.e. 30pts of the test) require you to write code in Racket and similar to what we did on the quiz, you just fill in the appropriate section of a .rkt file provided on bbLearn (and you also turn it on bbLearn). The bulk of the midterm (70pts, the first 8 questions) you just write on a pdf available in Gradescope and submit it there, just like the labs. It can count as a star towards your overall course grade if you get an A on it (90% or higher) -- but don't panic: like all timed assessments, corrections will be permitted (but you are strongly encouraged not to wait any longer than one week after they are returned).

Note that Lab#11, which would normally done during that time, will be posted at the end of the prior week (Fri July 28th) and can be done individually or in groups and won't be due until the end of the following week (11:59pm on Fri Aug 4th). The material of Lab#11will NOT be covered on the midterm.

Here is the list of topics you should be comfortable with for the midterm (in chronological order, not necessarily order of importance). Feel free to discuss in the #reviewing Discord channel.

- basic Racket commands, especially knowing the difference between rest/second, and cons/append/list
- building a list using only cons and null
- extracting a member from a list using only first/rest
- writing recursive functions, predicates, and contracts
- equational reasoning in Racket
- working with abstractions (Pnums, Dubnums, PolyLists etc)
- higher order functions: especially map, foldr, λ
- from an expression, make its truth table, and vice versa (esp understand the implies operation and the concept of being "vacuously true")
- proving expressions using Boolean Algebra rules (the list of rules will be provided)
- Normal forms (especially applications of DNF/CNF)
- using DIMACS in SAT Solvers

Listed on the next page are a couple good review question for the midterm. Feel free to discuss them on the #reviewing channel in our class Discord. Do not use any functions except those taught in class

[a] implement a predicate that checks whether an input is a dubNum. (note: do not use the length function in your answer)

```
#|
Input contract: L is anything
output contract: (dubNum? L) is equivalent to saying "L is a dubnum".  In other words, it is True if and only if L is a dubnum
Examples:  (dubNum? 'z), (dubNum? '(DP1 z)),  (dubNum? '(D (DP1 z))) would all output #t. and (dubNum? 0), (dubNum? null), (dubNum? '(D z)), (dubNum? '(D DP1 z)) would all output #f
|#

(define/contract (dubnum? L) (-> any/c boolean?)

…..)
```

[b] Implement a function called `change01` according to the following specs. You do not need to implement the contracts for this question.

```
; input contract:  L is a nonempty list of 0s and 1s, and k is a positive integer
; output contract: (change01 k L) is the same list as L except that the kth 0 has been changed
                   into a 1.  If there is no kth zero, then L is returned.
; example:  (change01 3 '(1 0 0 1 0 1 0 1))  would output '(1 0 0 1 1 1 0 1)

(define (change01 L)
…. )
```