For this exam, the Racket language level is **Beginning Student**. You may find these Racket functions helpful. Note that you may not need to use all of them in your solutions for the midterm.

- `(cons v lst)` constructs a list with first element `v` followed by list `lst`.
- `(first lst)` produces the first element in the non-empty list `lst`.
- `(rest lst)` produces a list that contains all elements of the non-empty list `lst` except the first element.
- `(length lst)` produces the number of elements of the list `lst`.
- `(empty? lst)` produces true if `lst` is an empty list and false otherwise.
- `(substring s a b)` produces a string consisting of the characters in `s` from positions `a` to `b-1`.
- `(substring s a)` produces a string consisting of the characters in `s` from positions `a` to the end of the `s`.
- `(string-append s1 s2)` produces a string consisting of the characters in `s1` followed by all of the characters in `s2`.
- `(string-length str)` produces the number of characters in the string `str`.
- `(string->list str)` produces a new list of characters corresponding to the content of `str`.
- `(list->string lst)` produces a new string corresponding to the characters contained in `lst`.
- `(number->string n)` produces the string version of the number `n`.
- `(integer? n)` produces true if `n` is an integer, false otherwise.
- `(number? x)` produces true if `x` is a number, false otherwise.
- `(symbol? s)` produces true if `s` is a symbol, false otherwise.
- `(odd? n)` produces true if `n` is an odd integer, false otherwise.
- `(even? n)` produces true if `n` is an even integer (including 0), false otherwise.
- `(zero? n)` produces true if `n` is the number 0, false otherwise.
- `(equal? x y)` produces true if `x` and `y` have the exact same value, false otherwise.
- `(quotient x y)` produces the integer result of dividing `x` by `y`.
- `(remainder x y)` produces the remainder of dividing `x` by `y`.
- `(sqr x)` produces the square of `x`.
- `(sqrt x)` produces the square root of `x`.
- `(max n1 n2)` produces the larger value when comparing `n1` and `n2`. There may be more than two arguments to the function.
- `(min n1 n2)` produces the smaller value when comparing `n1` and `n2`. There may be more than two arguments to the function.
- `(posn-x p)` produces the x-coordinate value of the built-in `Posn` structure `p`.
- `(posn-y p)` produces the y-coordinate value of the built-in `Posn` structure `p`.
- `(make-posn a b)` produces a `Posn` with x-coordinate `a` and y-coordinate `b`.
- `(posn? item)` produces true if `item` is a `Posn`, false otherwise.
- `(check-expect act exp)` tests whether `act` and `exp` are equal.
- `(check-within act exp tol)` tests whether |act - exp| <= `tol`