CS 115 Fall 2010

Assignment 3
Due 4:00 PM on Monday, October 18

For this and all subsequent assignments, you are expected to use the design recipe when writing functions from scratch. Do not copy the purpose directly from the assignment description. The purpose should be written in your own words and include reference to the parameter names of your functions. The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources. Test data for all questions will always meet the stated assumptions for consumed values.

Please read the course Web page for more information on assignment policies and how to organize and submit your work. Be sure to download the interface file from the course Web page and to follow all the instructions listed in the style guide (on the Web page). Specifically, your solutions should be placed in files a3qY.rkt, where Y is a value from 1 to 4. For full marks, it is not sufficient to have a correct program. Be sure to follow all the steps of the design recipe, including the definition of constants and helper functions where appropriate.

Language level: Beginning Student.
Coverage: Module 3
Useful structure and data definitions:

(define-struct contact (name email-address))
;; A **contact** is a structure *(make-contact n e)* where
;;   *n* is a non-empty string representing a person's name and
;;   *e* is a non-empty string representing a person's email address.

(define-struct email (from to subject message))
;; An **email** is a structure *(make-email f t s m)* where
;;   *f* is a contact representing who sent the email,
;;   *t* is a contact representing who received the email,
;;   *s* is a string representing the subject of the email, and
;;   *m* is a non-empty string representing the text of the message.

(define-struct card (value suit))
;; A **card** is a structure *(make-card v s)* where
;;   *v* is an integer in the range from 2 to 11,   *** note this range is different from the
;;   range given in the course slides
;;   *s* is a symbol from the set 'hearts, 'diamonds, 'spades, and 'clubs.

(define-struct gps-point (lat long alt))
;; A **gps-point** is a structure*(make-gps-point d1 d2 a)* where
;;   *d1* is a real number representing a latitude position measured in degrees
;;   *d2* is a real number representing the longitude position measured in degrees, and
;;   *a* is a real number representing an altitude measured in metres.

1. A convex quadrilateral (all interior angles are less than 180 degrees) can be defined by identifying four distinct points on the Cartesian plane: **a**, **b**, **c**, and **d**. The sides of the quadrilateral are **ab**, **bc**, **cd**, and **ad**. The side **ab** is opposite the side **cd**, and the side **bc** is opposite the side **ad.** Certain quadrilaterals have special features and hence have special names. Write a function called `identify-quad` that consumes four `posn` values and determines which category **best** describes the quadrilateral the points form.
   - A square – all sides are the same length, and all sides meet at right angles.
   - A rhombus – all sides are the same length.
   - A rectangle – both pairs of opposite sides are parallel, and all sides meet at right angles.
   - A parallelogram – both pairs of opposite sides are parallel.
   - A trapezoid – one pair of opposite sides is parallel.
   - Anything else is simply a quadrilateral.

   Your function consumes the points identifying the corners of the quadrilateral (`a`, `b`, `c`, `d`), and produces a symbol, one of: `'square`, `'rhombus`, `'rectangle`, `'parallelogram`, `'trapezoid`, or `'quadrilateral`. This list is in order from most restrictive to least restrictive category. If the points consumed meet the requirements of more than one of the categories, the function should produce the symbol describing the most restrictive choice. For example `(identify-quad (make-posn 0 0) (make-posn 1 4) (make-posn 5 4) (make-posn 4 0))` produces `'parallelogram`.

   Hints:
   - Two lines are parallel if the slopes of the lines are the same. The slope of the line that passes through the points (x_1, y_1) and (x_2, y_2) is ($y_2 - y_1$)/($x_2 - x_1$). Recall that any two points that have the same x-value form a line with an infinite slope.
   - Two lines form a right angle if the slopes of the lines are the negative reciprocal of each other. For example if the slope of one line is ½ and the slope of another line is -2, then these lines are perpendicular (at right angles). The negative reciprocal of 0 is infinity.
   - To determine if a quadrilateral is a square or a rectangle, it is enough to determine if one of the corners meets at a right angle, as long as the other restrictions are true.

   You may use the `distance` function from the course notes in your solution, however you should note in the comments for that function where the code originated.

2. Write a function called `reply` that consumes an `email` and a string and produces an `email` that is a reply to the `email` consumed using the message given. The `contact` that sends the reply email will be the same as the `contact` that received the original `email`. The `contact` that receives the reply `email` will be the same as the `contact` that sent the original `email`. The subject of the reply `email` will be the same as the subject of the original `email` except that it will start with `"RE: "`. The `message` of the reply `email` will be the string consumed by the `reply` function, followed by a space, followed by the `name` of the sender, followed by a space, followed by the word `"said"` followed by a space followed by the original message.

   For example `(reply (make-email (make-contact "Sandy Graham" "slgraham@uwaterloo.ca") (make-contact "J.P. Pretti" "jpretti@uwaterloo.ca") "Hello" "I love CS115!") "Me too.")` produces `(make-email (make-contact "J.P. Pretti" "jpretti@uwaterloo.ca") (make-contact "Sandy Graham" "slgraham@uwaterloo.ca") "RE: Hello" "Me too. Sandy Graham said I love CS115!")`

3. The card game thirty-one is played using three cards in each player's hand. Each player takes a turn by picking up one card and discarding one card, trying to get a total as close to 31 as possible. Each player always has exactly three cards. When counting the score of a hand, cards must be the same suit. The best possible score for a hand with three cards is 31 points. Aces have a value of 11, face cards (king, queen, or jack) and tens have a value of 10, and the other cards count as their given numeric value. (Note, there are no symbols used to represent these cards.) Write a function called `score-31` that consumes three `cards` and produces the maximum score the three cards can make. For example `(score-31 (make-card 11 'spades) (make-card 9 'spades) (make-card 10 'spades))` produces 30, and `(score-31 (make-card 8 'diamonds) (make-card 10 'clubs) (make-card 7 'diamonds))` produces 15.

4. All geographic points on earth can be described by their latitude, longitude and altitude. These points are used in GPS (global positioning system) devices to help with navigation. Latitude is a number in the range [-90, 90], where positive numbers represent positions north of the equator (i.e. the northern hemisphere) and negative numbers represent positions south of the equator (i.e. the southern hemisphere). Longitude is a number in the range [-180, 180] where positive numbers represent positions east of Greenwich, England (i.e. eastern hemisphere) and negative numbers represent positions west of Greenwich (i.e. western hemisphere). Note that -180 longitude and 180 longitude are actually the same place on Earth. Note that 0 degrees latitude is the equator – it is the division between the northern and southern hemisphere, but not considered part of either. Similarly, 0 degrees longitude and 180/-180 degrees longitude are division lines for the eastern and western hemispheres, but not considered part of either. Altitude is a number of metres above or below sea-level.

   Write a function called `where-in-the-world` that consumes a `gps-point` and produces a symbol indicating where in the world that point is located given the rules listed below.

The function will produce one of `'Africa`, `'Antarctica`, `'Asia`, `'Australia`, `'Europe`, `'NorthAmerica`, `'SouthAmerica`, or `'Ocean`.

- Any point with an altitude of less than or equal to 0 is considered in the ocean. All other points are considered land.
- Antarctica is any land south of and including -60 degrees latitude.
- Australia is any land that is not Antarctica that is in the southern and eastern hemispheres, south of and including -10 degrees latitude and east of and including 110 degrees longitude.
- Africa is any land south of and including 37 degrees latitude, east of and including -10 degrees longitude and west of and including 63 degrees longitude.
- Europe is any land in the northern hemisphere, east of and including -28 degrees longitude, west of and including 68 degrees longitude and north of Africa.
- North America is any land in the western hemisphere, that is not Europe, and north of and including 5 degrees latitude.
- South America is any land in the western hemisphere, that is not North America, Africa, or Antarctica.
- Asia is any land that is not part of one of the other continents.

For example `(where-in-the-world (make-gps-point 10 -5 0))` produces `'Ocean` and `(where-in-the-world (make-gps-point -10.5 110.5 123.4))` produces `'Australia`.