# CMPE 100 – Mid-term Exam 2 – 2013-2014 Fall

In each of the questions below write the function to produce the described output according to problem statement. Make sure to include cases in which program must produce an error. Also put test cases (i.e. `check-expect` tests) in your programs. You need to add documentation comments only if you find them necessary to describe your approach.

1. **(15 points)** Following is a data structure definition which represents a point in a two dimensional coordinate system:
   ```
   (define Point (x y))
   ```
   Write a Racket program which implements a function that takes two instances of Point structure and returns the distance between the two points. Create two such instances and give an example call of your function.
   NOTE: Distance between two points, ($x_1$,$y_1$) and ($x_2$,$y_2$) is found as $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$ .
   You may use the `sqrt` function inRacket to compute square-root.

**ANSWER:**
```
(define-struct Point (x y))
(define (distance point1 point2)
  (sqrt (+
        (sqr (- (Point-x point1) (Point-x point2)))
        (sqr (- (Point-y point1) (Point-y point2))))))
(check-expect (distance
              (make-Point 0 0)
              (make-Point 1 0)) 1)
```

2. **(15 points)** Write a Racket function which takes an integer number as a parameter and returns the following function value:

$$f(n)=\begin{cases} undefined & if\, n\leqslant 0 \\ 1 & if\, n=1 \\ \dfrac{n}{f(n-1)} & otherwise \end{cases}$$

**ANSWER:**
```
(define (f n)
  (cond
    ((<= n 0) (error "undefined"))
    ((= n 1) 1)
    (else (/ n (f (- n 1))))))
(check-expect (f 2) 2)
```

3. **(20 points)** Write a Racket function which takes a positive integer number as a parameter and returns the following sum:

$$f(n)=\sum_{i=1}^{i=n}\frac{1}{i}=\frac{1}{1}+\frac{1}{2}+\ldots+\frac{1}{n}$$

Make sure to provide a recursive formulation of the computation in your answer.

**ANSWER:**

**The recursive formulation would be as follows**

$$f(n)=\begin{cases} undefined & if\ n\leqslant 0 \\ 1 & if\ n=1 \\ f(n-1)+\frac{1}{n} & otherwise \end{cases}$$

```
(define (f n)
  (cond
    ((< n 1) (error "undefined"))
    ((= n 1) 1)
    (else (+ (/ 1 n) (f (- n 1))))))
(check-expect (f 2) 1.5)
```

4. **(25 points)** Write a Racket function which takes a list of numbers as its parameter, and returns a list which contains only the negative numbers in the list. For example:

```
(negatives (list 2 0 -4 1 -1)) ; returns (list -4 -1)
```

**ANSWER:**

```
(define (negatives list)
  (cond
    ((empty? list) empty)
    ((< (first list) 0)
     (cons (first list) (negatives (rest list))))
    (else (negatives (rest list)))))
(check-expect (negatives (list 2 0 -4 1 -1))
              (list -4 -1))
```

5. **(25 points)** Write a Racket program which implements the following function:

$$f(n)=\begin{cases} undefined & if\ n\leqslant 0 \\ 1 & if\ n=1 \\ \frac{n\cdot f(n-1)}{n+f(n-1)} & otherwise \end{cases}$$

Make sure to **consider efficiency of your implementation**, e.g. avoid double recursion!

**ANSWER:**

```
(define (helper n fn-1)
  (/ (* n fn-1) (+ n fn-1)))
(define (f n)
  (cond
    ((<= n 0) (error "undefined"))
    ((= n 1) 1)
    (else (helper n (f (- n 1))))))
(check-expect (f 2) 2/3)
```