İstanbul Bilgi University

Department of Computer Engineering

CMPE 100: Introduction to Computing

2019/2020 Fall - Midterm Quiz (Duration: 60 minutes)

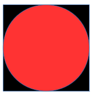
PARTA	PARTB	PARTC					TOTAL

Student Name: ___________________________**Student Number:** _____________

PART A: (TOTAL 16 points) Answer the following questions by giving the value and data type resulting from the given expression. Some examples are given in the first two lines:

		VALUE	**DATA TYPE**
	`(square 100 "solid" "black")`	A square of size 100	Image
	`(+ 3 5)`	8	number
1	`(/ (+ 7 5) (* 2 2))`	3	number
2	`(or (< 4 3) (> 3 2))`	#true	boolean

3	`(cond`
	`((> 3 2) "a")`
	`((< 3 5) "b")`
	`(else "n"))`
		"a"	string

4	`(overlay`
	`(circle 50 "solid" "red")`
	`(square 100 "solid" "black"))`
			image

5	`(image-height`
	`(circle 100 "solid" "black"))`
		200	number

6	`(and (< 4 3) (> 3 2))`	#false	boolean

7	`(cond`
	`((< 2 1) 1)`
	`((< 2 3) (< 2 3))`
	`(else "a"))`
		#true	boolean

8	`(cond`
	`((and #false (< 2 3)) 3)`
	`((< 2 3) 4)`
	`(else 5))`
		4	number

PART B: (TOTAL 18 points) Examine each of the short programs below. **Each program only deals with the data type “number”.** For each, find the mistake(s) in the following program, or if there are no mistake indicate that. Some examples are given in the first two lines:

		Indicate if correct, or describe the mistake(s)

	`(define (f x)`
	`        (+ x y))`
		y is not defined anywhere in the program

	`(define (f x)`
	`  (* x x))`
		No mistakes

1	`(define (f x y z)`
	`   (cond`
	`      ((and (<= x z) (+ x y)) x)`
	`      (else    y)))`
		and: question result is not true or false and accepts a boolean value

2	`(define (f x n)`
	`  (cond`
	`    ((= n 0) 1)`
	`   (else (f (- n 1)))))`
		f: expects 2 arguments, but found only 1

3	`(define (f x)`
	`  x)`
		No mistakes

4	`(define (f n)`
	`  (* 2 (f (- n 1))))`
		There is no termination condition

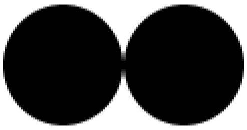
5	`(define (f x y)`
	`  (cond`
	`      ((> y x) y)`
	`      ((> x y) x)))`
		cond: all question results were false when x and y are equals

6	`(define (f x n)`
	`  (cond`
	`     ((= n 0) 1)`
	`     (* x (f (- n 1)))))`
		else or condition
		f: expects 2 arguments, but found only 1

PART C: (TOTAL 66 points in 4 problems) In each of the questions below apply the **design recipe** to write the function to produce the described output according to problem statement. Write your programs in the space given below each question, not on a separate answer sheet. You need to add documentation comments only if you find them necessary to describe your approach.

1. (13 points) Write a Racket function named **`twice`** which takes an image as a parameter, and returns an image which contains two copies of the given image side by side. For example:

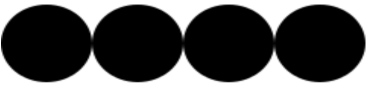
`(twice (circle 30 "solid" "black")) →`



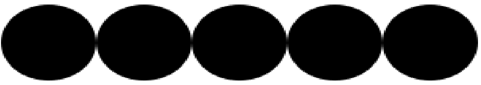
```
;constant:
(define ball (circle 10 "solid" "red"))
(define table (square 20 "solid" "black"))

;contract: twice : image --> image
(check-expect (twice ball) (beside ball ball))
(check-expect (twice table) (beside table table))

(define (twice img)
  (cond
  ((not (image? img))(error "not an image"))
  (else (beside img img))))

(twice ball)
```

2. (13 points) Write a Racket function named **ntuple** which takes an image and a positive integer, ***n***, as parameters, and returns an image which contains ***n*** copies of the given image side by side. For example:

`(ntuple (circle 30 "solid" "black") 4) →`



`(ntuple (circle 30 "solid" "black") 5) →`



```
;contract: ntuple : image number ---> image
(check-expect (ntuple ball 5) (beside ball ball ball ball ball))
(check-expect (ntuple table 2)(beside table table))

(define (ntuple img n)
  (cond
  ((not (image? img))(error "not an image"))
  ((<= n 0) (error "must be positive integer"))
  ((= n 1) img)
  (else (beside img (ntuple img (- n 1))))))

(ntuple ball 5)
(ntuple table 10)
```

3. (15 points) What is the output of the following code?

```
(define (f m n)
(cond
((= m n) m)
((> m n) (f (- m n) n))
(else (f m (- n m)))))

(f 24 18)  → 6
(f 5 7) → 1
```

4. (15 points) Write a Racket function named **myPower** which takes a number,***x*** and a non negative integer, ***n*** as parameters, and computes **x^n .**

```
(myPower 2 3) → 8 (2*2*2)
(myPower 25 0) → 1
```

;contract: myPower: number number --> number

```
(check-expect (myPower 2 3) 8)
(check-expect (myPower 2 0) 1)
(check-expect (myPower 2 1) 2)
(check-expect (myPower 1 3) 1)

(define (myPower x n)
 (cond
 ((or (< x 0) (< n 0)) (error "must be positive integer"))
 ((= n 0) 1)
 ((= n 1) x)
 ((= x 1) 1)
 (else (* x (myPower x (- n 1))))))

(myPower 2 3)
(myPower 2 1)
```

5. (10 points) Write a Racket function named **f** which takes a positive integer, ***x*** as parameters, and returns the following function value:

$$f(x)=\begin{cases} undefined & if\ x \leqslant 0 \\ x & if\ x=1 \\ (x-1)*f(x-1) & otherwise \end{cases}$$

;contract: f : number --> number

```
(check-expect (f 2) 1)
(check-expect (f 3) 2)

(define (f x)
  (cond
  ((<= x 0) (error "must be positive and nonzero"))
  ((= x 1) x)
  (else (* (- x 1)(f (- x 1))))))

(f 3)
(f2 4)
```