B

FAILURE TO COMPLETELY FILL THE TOP PART OF THIS PAGE WILL LOSE 2%

THE UNIVERSITY OF BRITISH COLUMBIA
CPSC 110: MIDTERM 1 - JANUARY 30TH, 2011

Name: _____ UBC Student #:_____ CS Dept. ID #:_____

Signature: _____

Lab Section:_____ Lecture Section (circle): 201 Gregor 202 Jim

Important notes about this examination

1. **This exam has 2 separate parts. You have 90 minutes to complete both parts.**
2. Except for questions 1 and 2, this exam will be graded largely on how well you **follow the design recipes**. You have been given a copy of the Recipe Exam Sheet. Use it!
3. Put away books, papers, laptops, cell phones... everything but pens, pencils, erasers and this exam.
4. Good luck!

Rules Governing Formal Examinations

1. Each candidate must be prepared to produce, upon request, a UBCcard for identification.
2. ~~Candidates are not permitted to ask questions of the invigilators, except in cases of supposed errors or ambiguities in examination questions~~. No questions will be answered in this exam. If you see text you feel is ambiguous, make a reasonable assumption, write it down, and proceed to answer the question.
3. No candidate shall be permitted to enter the examination room after the expiration of one-half hour from the scheduled starting time, or to leave during the first half hour of the examination.
4. Candidates suspected of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action:
    - having at the place of writing any books, papers or memoranda, calculators, computers, sound or image players/recorders/transmitters (including telephones), or other memory aid devices, other than those authorized by the examiners;
    - speaking or communicating with other candidates; and
    - purposely exposing written papers to the view of other candidates or imaging devices. The plea of accident or forgetfulness shall not be received.
5. Candidates must not destroy or mutilate any examination material; must hand in all examination papers; and must not take any examination material from the examination room without permission of the invigilator.
6. Candidates must follow any additional examination rules or directions communicated by the instructor or invigilator.

Please do not write in this space:

Question	Grader	Mark
6B		
EC		

Problem 6B - Completing a World Program

In this problem you will complete the design of the world program you started in problem 6B. To help you do that we have started a program file, defined constants, defined a data definition for changing world state and defined a main function. We have also defined the wish list entries for the two handler functions required. Our constants may be different than the ones you defined in 6A. Do not worry about that, for this part of the problem use our constants as defined below.

You must complete the design of the grow function. Completing render is extra credit.

```
(require 2htdp/image)
(require 2htdp/universe)

;; Watching grass grow!


;; Constants:

(define WIDTH  200)
(define HEIGHT 300)

(define GRASS-WIDTH 2)

(define CTR-X (/ WIDTH 2))

(define SPEED 2) ;number of pixels grass grows each tick


(define MTS (empty-scene WIDTH HEIGHT))


;; Data definitions

;; Grass is Natural
;; interp. the length of the blade of grass in pixels
(define G1 0)
(define G2 10)

#;
(define (fn-for-grass g)
  (... g))

;; Template rules used:
;; - atomic distinct: Natural
```

```
;; Functions

;; Grass -> Grass
;; start the grass growing show; run with (main 0)
;; <no tests for main functions>
(define (main g)
  (big-bang g
            (on-tick grow)     ; Grass -> Grass
            (to-draw render))) ; Grass -> Image
```

Complete the design of grow.

```
;; Grass -> Grass
;; grow the grass by SPEED
;; !!!

;(define (grow g) 1)

[ 2 points for 1 correct check-expect
[ but only 1 point if it uses 2 in the expected value rather SPEED
[ 1 point for correct function header
[ 1 point for clearly follows template
[ 1 point for correct body, only if uses constants
[ -1 if * instead of +

(check-expect (grow 2) (+ 2 SPEED))

(define (grow g)
  (+ g SPEED))
```

Completing the design of render is extra credit. Because it is extra credit it will need to be almost entirely correct to get marks. But here is a hint: `(place-image/align <image> <x> <y> "center" "bottom" MTS)` will put the bottom center of <image> at position <x> <y> on MTS.

```
;; Grass -> Image
;; render grass on MTS
;; !!!

;(define (render g) MTS)

[ 3 points for correct test
[   but only 1 if it doesn't use the constants properly
[ 3 points for good function definition
[   but only 1 if it doesn't use the constants properly
[ -2 if missing args to rectangle
[ -4 if */+ SPEED
[ -4 if place-image instead of place-image/align

(check-expect (render 4)
  (place-image/align (rectangle GRASS-WIDTH 4 "solid" "green")
                     CTR-X
                     HEIGHT
                     "center"
                     "bottom"
                     MTS))

(define (render g)
  (place-image/align (rectangle GRASS-WIDTH g "solid" "green")
                     CTR-X
                     HEIGHT
                     "center"
                     "bottom"
                     MTS))
```