; ● CSC104 Winter 2020 —Exercise #4 — Print out and fill in by hand, then hand in to the TA at the start of your quiz. ●

; UTorID (login ID) :

; Surname :

; Given Name :

; ● Part I.

; Define sesqui so that it behaves as shown :

```
(step (sesqui 1903))
;  ... produces the steps ...
(+ 1903 150)
2053
```

```
(step (sesqui 2020))
;  ... produces the steps ...
(+ 2020 150)
2170
```

; Beside each of these two expressions write its value : `sesqui` `(sesqui 1815)`

; Show, with standard underlining, the following steps ...

```
(step (map sesqui (list 1815 1906 1903)))
```

```
(step (hide sesqui) (map sesqui (list 1815 1906 1903)))
```

; Define ! so that it behaves as shown ...

```
(step (! "wow"))
;  ... produces the steps ...
(text-join "wow" "!")
"wow!"
```

```
(step (! "whatever"))
;  ... produces the steps ...
(text-join "whatever" "!")
"whatever!"
```

; Beside each of these two expressions write its value : `!` `(! "buddy")`

; Show, with standard underlining, the steps for : `(step (hide !) (map ! (list "wow" "whatever" "buddy")))`

; Define born-1906? so that it behaves as shown ...

```
(step (born-1906? (list 1906 "Goedel" "Kurt")))
;  ... produces the steps ...
(same? (first (list 1906 "Goedel" "Kurt")) 1906)
(same? 1906 1906)
#true
```

```
(step (born-1906? (list 1815 "Lovelace" "Ada")))
;  ... produces the steps ...
(same? (first (list 1815 "Lovelace" "Ada")) 1906)
(same? 1815 1906)
#false
```

; Beside each of these two expressions write its value ...

```
born-1906?
```

```
(born-1906? (list 1903 "Church" "Alonzo"))
```

; Define text-first so that it behaves as shown :

```
(step (text-first "ruby"))
;  ... produces the steps ...
(first (text->list "ruby"))
(first (list "r" "u" "b" "y"))
"r"
```

```
(step (text-first "jade"))
;  ... produces the steps ...
(first (text->list "jade"))
(first (list "j" "a" "d" "e"))
"j"
```

; Beside each of these two expressions write its value ...

```
text-first
```

```
(text-first "onyx")
```

; Show, with standard underlining, the steps for ...

```
(step (hide text-first)
      (map text-first (list "ruby" "jade" "onyx")))
```

```
(step (map text-first (list "ruby" "jade" "onyx")))
```

; ● Part II. Assume the following definitions have been entered/run ...

```
(define R      (random 1000000))
(define (r  _) (random 1000000))
(define (Rf _) R)
```

; ... then under each of these expressions write its value ...

```
(same? (random 1000000)
       (random 1000000))
```

```
(same? R R)
```

```
(same? (r "hmm")
       (r "hmm"))
```

```
(same? (Rf "hmm")
       (Rf "hmm"))
```

```
r
```

```
Rf
```

; ● Part III.

; Based on this definition ...

```
(define (A n)
    (if (same? n 0) △
        else (above (A (- n 1))
                    (wide (A (- n 1))))))
```

; ... show the steps, with standard underlining, for ...

```
(step (A 0))
```

```
(step (hide (A 0))
      (A 1))
```

```
(step (hide (A 1))
      (A 2))
```

; ● Part IV.

; Based on these definitions ...

```
(define C (circle 20))
```

```
(define (arrange an-image)
  (beside C (tall an-image) C))
```

; ... show the steps, with standard underlining, for ...

```
(step (arrange C))
```

; Beside each of these two expressions write its value :

```
C
```

```
arrange
```

; Based on this definition ...

```
(define (B k)
  (if (same? k 0) C
      else (arrange (B (- k 1)))))
```

; ... show the steps, with standard underlining, for ...

```
(step (B 0))
```

```
(step (hide (B 0)
            arrange)
      (B 1))
```

; Show the value of :

```
(B 2)
```