PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

Term test #1

CSC 104H1
Duration — 50 minutes

aids allowed: 8.5" x 11" handwritten aid sheet, both sides

Last Name: Heap

First Name: Danny

Do not *turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below.)

This test consists of 5 questions on 8 pages (including this one). *When you receive the signal to start, please make sure that your copy of the test is complete.*

Please answer questions in the space provided. You will earn 20% for any question you leave blank or write "I cannot answer this question," on. You will earn substantial part marks for writing down the outline of a solution and indicating which steps are missing.

Good Luck!

QUESTION 1. [5 MARKS]

Discuss why modern computers are powered by electricity rather than steam. What are the advantages and disadvantages of this choice?

electricity:

- cleaner
- smaller moving parts (microscopic for electronics)
- quieter
- faster
- less maintenance (wires vs rods + gears).
- less heat

steam: few advantages

- No humidifier needed ... :)

1 mark for each relevant point.

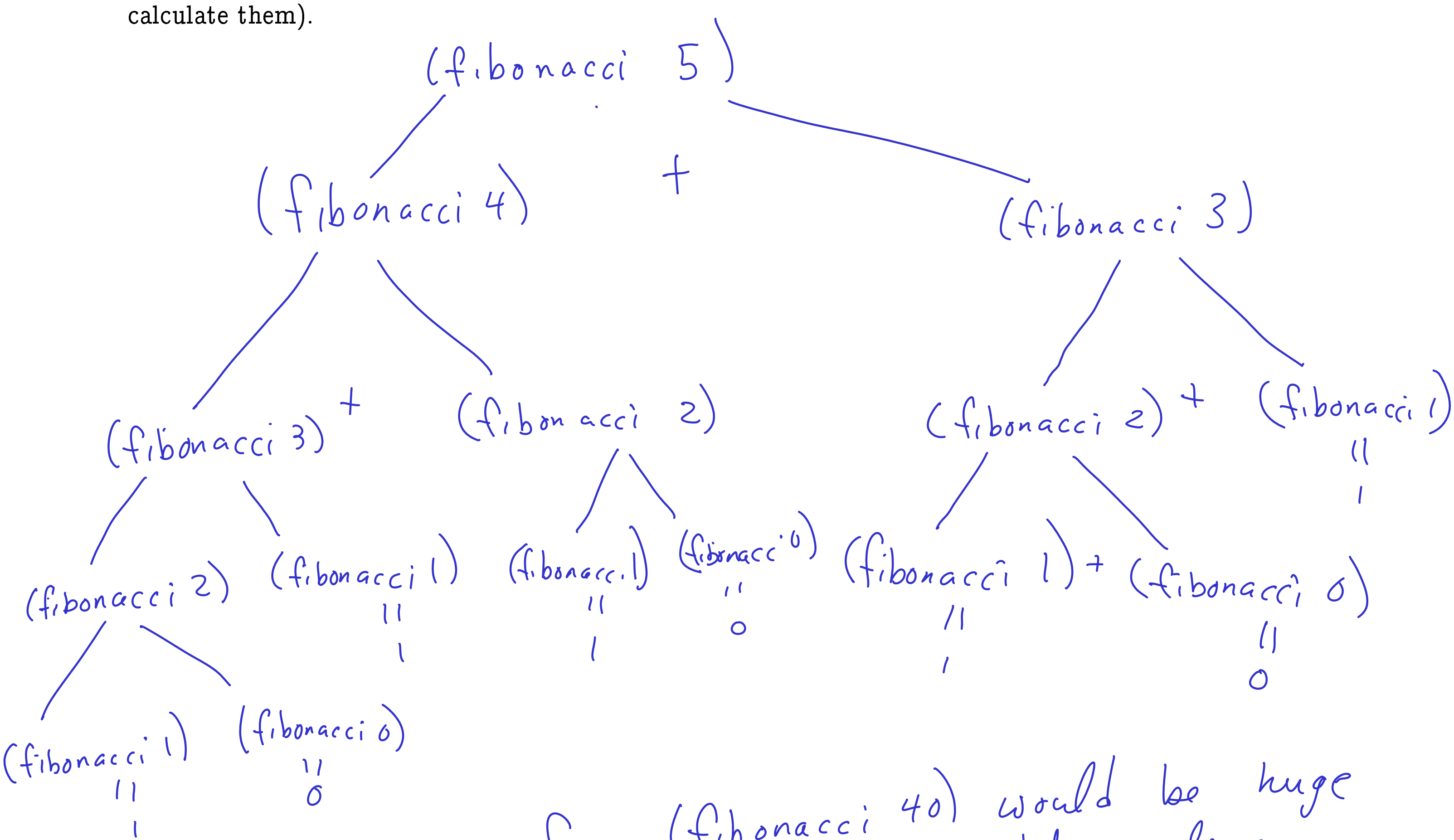
Question 2. [5 marks]

Note: in what follows, please feel free to ask about any DrRacket built-in features.

The 0th fibonacci number is 0. The 1th fibonacci number is 1. Larger fibonacci numbers are defined by the DrRacket code below to be the sum of their two predecessors:

```
; fibonacci : number -> number
; Produce nth fibonacci number
(define (fibonacci n)
  (if (< n 2)
      n
      (+ (fibonacci (- n 1))
          (fibonacci (- n 2)))))
;
(check-expect (fibonacci 2) (+ (fibonacci 1) (fibonacci 0)))
(check-expect (fibonacci 3) (+ (fibonacci 2) (fibonacci 1)))
```

Show all the steps to calculate (fibonacci 5) in this way. Explain why this takes even a computer a substantial amount of time for values as modest as (fibonacci 40) or (fibonacci 50) (please don't calculate them).



The corresponding tree for (fibonacci 40) would be huge with many repeated function calls, so it takes a long time.

QUESTION 3. [5 MARKS]

Suppose I have a 1024-sheet dictionary and I'm comparing two search techniques. `Linear search` requires, on average, 512 operations — in an average search I examine half of the pages, starting from the front and working to the back, before I find the sheet the word I'm searching for should be on. `Binary search` requires, on average, 10 operations — in an average search I split the stack of pages in half 10 times before I find the sheet the word I'm searching for should be on.

How many operations, on average, would `Linear Search` and `Binary Search` each require if the dictionary doubled to 2048 sheets? Explain your thinking.

Linear Search would require, on average, 1024 operations for a dictionary of 2048 sheets. On average, it examines half the sheets.

Binary Search would require, on average, 11 operations for a dictionary of 2048 sheets. 1 operation splits the stack of pages down to 1024 sheets, and then the same 10 operations as before.

Question 4. [11 marks]

Assume the expressions below have been typed into the definitions pane of DrRacket. Below each parenthesized expression write, draw, or describe its effect when the "Run" button is clicked.

```
(require picturing-programs)
```

this provides funtions for working on images.

21

7

```
(rotate (* 3 (string-length "fifteen")) pic:hacker)
```

Produce a hacker image rotated 21 degrees ccw.

30

9

```
(overlay
 (triangle (+ 21 (string-length "seventeen")) "solid" "green")
 (circle 40 "solid" "blue"))
```

Overlay a solid green triangle of size 30 on a solid blue circle of size 40.

3

```
(beside
 (scale (string-length "two") pic:calendar)
 pic:hacker)
```

Produce a 3X enlargement of calendar beside hacker

9

3

```
(+ 2 5 (* 3 (- 7 4)))
```

16

Produce number 16

false

4

4

```
(or
 (> (string-length "five") (string-length "four"))
 (< 7 10))
```

true

Produce true.

Question 5. [10 marks]

For each of the two functions square-string-length and rotate-stack, I have provided a summary sentence, one check-expect expression, started the define statement, and written an incomplete contract comment (I left out what is consumed and what is produced, after the ":"). You should complete both functions by:

(i) Adding another appropriate check-expect expression for each function

(ii) Completing the define statement with the body of the definition for each function

(iii) Completing the contract comment, saying what is consumed and what is produced for each function

```
(require picturing-programs)
                           ?
; square-string-length :  string -> image
; Produce solid red square with size string-length of s
(define (square-string-size s)
  ; write the body of the definition below here
    (square (string-length s) "solid" "red")




)
; write your check-expect below this one
(check-expect (square-string-size "seventeen")
              (square 9 "solid" "red"))
(check-expect (square-string-size "word")
              (square 4 "solid" "red"))
```

```
; rotate-stack : image → image
; Produce im above 45 degree rotated version of itself
(define (rotate-stack im)
  ; write the body of the definition below here
   (above im (rotate 45 im))




)
; write your check-expect below this one
(check-expect (rotate-stack pic:hacker)
              (above pic:hacker (rotate 45 pic:hacker)))
 (check-expect (rotate-stack pic:calendar)
                         (above pic:calendar (rotate 45 pic:calendar))
```

1: _______/ 5

2: _______/ 5

3: _______/ 5

4: _______/11

5: _______/10

TOTAL: _______/36