QUESTION 1. [5 MARKS]

Contrast the way computers store information with the way computer users view the same information. Give three examples.

SAMPLE SOLUTION: Computers store numbers as sequences of high and low voltages in memory, whereas human users view them as sequences of digits in their favourite number base (usually base 10). A computer might store an image as several sectors of varying magnetic dipoles on a hard drive, whereas a human user views it as a single visual rectangle. A computer might store the next great Canadian novel as several sectors comprised of pits on the surface of a compact disk, whereas a human user will view this a a single collection of text characters.

QUESTION 2. [5 MARKS]

The Von Neumann bottleneck is due to moving data and instructions from RAM memory to the CPU over a 32-bit or a 64-bit bus. Why don't computer designers simply put all the memory directly on the CPU?

SAMPLE SOLUTION: Although the transistors that make up RAM are extremely small, they do occupy some space, so if you have billions of them, some of them will be (relatively) far from the portions of the CPU that need them. If you make them all equally accessible, you will have no extra-fast memory, such as the registers IR, PC, and R1, to perform frequently-needed, intermediate tasks. Also, the more transistors you pack close to the CPU, the more heat you will need to dissipate.

QUESTION 3. [9 MARKS]

PART (A) [4 MARKS]

Convert the numbers 13 and 5 to binary (base 2). Show how you do it.

SAMPLE SOLUTION: There are several ways to do this. Here's one.

Since 13 is odd, I know that its units digit is 1, and that removing the units digit from the binary representation of 13 is like dividing by 2 and rounding down to 6. Now I have to find the binary representation of 6, which I know is even, so it has units digit 0. For the remaining digits, I divide 6 by 2 and round down to 3, which is odd so the units digit is 1. To find the remaining digits I divide 3 by 2 and round down to 1, which is odd so another 1. Finally I divide 1 by 2 and round down to 0, and I'm done — zeros on the left make no difference, and this result will repeat forever.

Putting it all together gives me 1101

A similar process with 5: it's odd so I have a units digit 1. Then I divide 5 by 2 and round down to 2, which is even so it contributes a zero. then I divide 2 by 2 and round down to 1, also odd (and I'm done). Putting it all together gives me 101

PART (B) [2 MARKS]

Add your binary representations of 13 and 5 IN BINARY. Show your work (for example, when you carry a digit to the next column).

SAMPLE SOLUTION: Every time a column sums to more than 1, I have to carry, which I indicate with a superscript 1:

$$\begin{array}{cccccc} & {}^{1}1 & 1 & {}^{1}0 & 1 \\ + & & 1 & 0 & 1 \\ \hline 1 & 0 & 0 & 1 & 0 \end{array}$$

PART (C) [3 MARKS]

Multiply your binary representations of 13 and 5 IN BINARY. Show your work (for example, when you carry a digit to the next column).

SAMPLE SOLUTION: I multiply 1101 by 101 digit-by-digit, then add the results in binary:

$$\begin{array}{ccccccc} & & & 1 & 1 & 0 & 1 \\ & & \times & & 1 & 0 & 1 \\ \hline & & & {}^{1}1 & 1 & 0 & 1 \\ & & {}^{1}0 & 0 & 0 & 0 & \\ & {}^{1}1 & 1 & 0 & 1 & & \\ \hline 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline \end{array}$$

QUESTION 4. [12 MARKS]

Suppose the following commands are entered in the definitions pane of DrRacket, and then the "run" button is pushed:

```
(require picturing-programs)

; a shape is a string, an image, and 2 integers
; interp. name, picture, interior angles, number of sides
(define-struct shape (name pic angles sides))

(define shapeA (make-shape "square" (square 10 "solid" "blue") 90 4))
(define shapeB (make-shape "triangle" (triangle 10 "solid" "green") 60 3))
(define shapeC (make-shape "pentagon" (regular-polygon 10 5 "solid" "red") 108 5))
(define shapeD (make-shape "hexagon" (regular-polygon 10 6 "solid" "green") 120 6))

(define L (list shapeA shapeD shapeB shapeC))

(define (add3 num) (+ num 3))
```

For each command below, explain, write, or draw what is produced if it is typed into the interactions pane of DrRacket. Please ask about any built-in DrRacket commands you like. `(add1 n)` adds 1 to `n`. `(first L)` produces the first element of `L`. `(rest L)` produces the sublist of `L` minus its first element. `(shape-name s)` produces the name part of `s`. `(regular-polygon 10 6 "solid" "green")` produces a green hexagon of size 10.

```
(map add3 (map add1 (list 1 3 5 7 9 11)))
```

SAMPLE SOLUTION: `(list 5 7 9 11 13 15)`

```
(first (rest (reverse L)))
```

SAMPLE SOLUTION: `(make-shape "triangle" . 60 3)`; where the dot is a green triangle.

```
(shape-pic (list-ref L 0))
```

SAMPLE SOLUTION: A blue square

```
(shape-name (first (reverse (rest L))))
```

SAMPLE SOLUTION: `"pentagon"`

```
(map sub1 (list 2 4 6 8 10 12))
```

SAMPLE SOLUTION: `(list 1 3 5 7 9 11)`

```
(map shape-angles (rest (reverse (rest L))))
```

SAMPLE SOLUTION: `(list 60 120)`

QUESTION 5. [8 MARKS]

Read the definition below.

```
; rs : string -> string
; No purpose statement!
(define (rs s)
  (cond
    [(> (string-length s) 1)
     (string-append (rs (substring s 1)) (substring s 0 1))]
    [else s]
    ))
```

`(string-length s)` produces the number of characters in string `s`. `(string-append s1 s2 ...)` produces a new string by appending strings `s1 s2 ...` from left to right. `(substring s 1)` produces the substring of `s` obtained by omitting its first character. `(substring s 0 1)` produces the length-1 string consisting of just the first character of `s`. Please ask about any built-in DrRacket functions you need to.

PART (A) [4 MARKS]

Write `check-expect` expressions for `(rs "e")` and `(rs "he")`.

SAMPLE SOLUTION:

```
(check-expect (rs "e") "e")
(check-expect (rs "he") "eh")
```

PART (B) [4 MARKS]

Explain step-by-step what happens when `(rs "the")` is run.

SAMPLE SOLUTION:

- `string-length` of "the" is greater than 1, so append `(rs "he")` to "t"
- `string-length` of "he" is greater than 1, so append `(rs "e")` to "h" to "t"
- `string-length` of "e" is not greater than 1, so append "e" to "h" to "t" — "eht"

PART (C) [2 MARKS]

Explain the purpose of `rs`.

SAMPLE SOLUTION: `(rs s)` produces a string with characters in reverse order from `s`

1: _____/ 5

2: _____/ 5

3: _____/ 9

4: _____/12

5: _____/ 8

TOTAL: _____/39