QUESTION 1. [5 MARKS]

In your opinion was government support essential to creating the internet? Explain why, or why not.

Solution: Yes. Creation of the internet required a huge investment in both infrastructure (long-distance cables, computers) and software (TCP/IP and other protocols). No profit was made until at least two decades after the invention of both the internet, and later html browsers. No private investors would be willing to wait 20+ years for return on their investment.

QUESTION 2. [5 MARKS]

Compare and contrast using a browser such as Firefox or Internet Explorer on a remote site, with using a terminal on a time-sharing computer in the 1960s.

Solution: Users of a 1960s-era terminal and a contemporary browser both share a resource --- a single computer in the 1960s case, services from a single host in the browser version --- "seamlessly," provided the host is able to switch between clients more quickly than the clients are able to generate demands. The contemporary browser experience is usually richer in media types (text, graphics, sound) and, in the best cases, faster (although stuff happens...) Not to mention being farther from the host computer.

QUESTION 3. [4 MARKS]

Carry out the following calculations between base ten (decimal) and base two (binary):

PART (A) [2 MARKS]

Convert base ten 57 to base two. Explain your procedure.

Solution: 57 is an odd number so I know that the right-most bit is a 1. I also know that the remaining bits represent 57/2 (rounded down) or 28. This is an even number, so I know the next bit is 0. The remaining bits represent 28/2, or 14. This is again an even number, so I know the next bit is a 0, and the remaining bits represent 14/2 or 7. This is an odd number, so I know the next bit is 1, and the remaining bits represent 7/2 (rounded down) or 3. An odd number again, so the next bit is a 1. The remaining bits represent 3/2 (rounded down) or 1. That's easy --- 1 in binary is 1, and we're done, once we gather up all the bits: 111001

CONT'D...

PART (B) [2 MARKS]

Convert base two 1110110 to base ten. Explain your procedure.

```
Solution: The right-most zero indicates that this number is twice
          base~two 111011, which has a 1 in the units column, a 1
          in the twos column, a 0 in the fours column, a 1 in the
          eights column, a 1 in the sixteens column, and a 1 in the
          thirty-twos column.  Summing up:
          (32 + 16 + 8 + 2 + 1) x 2 = 118
```

QUESTION 4. [10 MARKS]

Assume the expressions below have been typed into the definitions pane of DrRacket, preceded by `(require picturing-programs)`. Below each parenthesized expression write, draw, or describe its effect when the "Run" button is clicked.

1.
```
(apply * (map sqrt (list 1 4 9 16)))
```

   Solution: map sqrt over a list produces a list of square roots: (list 1 2 3 4), then applying * evaluates (* 1 2 3 4) which yields 24

2.
```
(circle
 (apply + (map string-length
               (list "how" "now" "brown" "cow"))) "solid" "green")
```

   Solution: map string-length over a list of strings yields a list of their lengths: (list 3 3 5 3), then applying + evaluates (+ 3 3 5 3), producing 14, finally (circle 14 "solid" "green") produces a solid green circle of radius 14.

3.
```
(first (rest (reverse (rest (list 0 1 2 3 4 5)))))
```

   Solution: The rest of (list 0 1 2 3 4 5) is (list 1 2 3 4 5), and the the reverse of this is (list 5 4 3 2 1), and the rest of this is (list 4 3 2 1), finally the first of this 4.

4.
```
(or
 (> (string-length "five") (string-length "four"))
 (equal? (length (list 1 2 4)) 3))
```

   Solution: string-length "five" is 4, and not greater than 4, the string-length of "four", so (> 4 4) evaluates to false. The length of (list 1 2 4) is 3, which is equal? 3, so (equal? 3 3) evaluates to true. Finally, (or false true) evaluates to true.

```
5. (apply <
          (map image-width
               (list (triangle 7 "solid" "green") (square 5 "outline" "blue"))))
```

Solution: map image-width over a list of two images produces a list of their widths, or (list 7 5). Applying < to this list produces (< 7 5), which is false.

QUESTION 5. [10 MARKS]

Read the definition of the mystery function (ss n) below. Assume that (require picturing-programs) has already been included above the definition.

```
; ss : number -> image
; No purpose statement!
(define (ss n)
  (cond
    [(zero? n) (square 10 "solid" "blue")]
    [else
     (beside (square (image-width (ss (- n 1))) "solid" "blue")
             (ss (- n 1)))]))
```

The contract says that (ss n) takes in a number and produces an image. (zero? n) produces true if n is 0, false otherwise. Please ask about any DrRacket functions you have to.

PART (A) [4 MARKS]

Write check-expect expressions for (ss 0) and (ss 1).

```
Solution: (check-expect (ss 0) (square 10 "solid" "blue"))
          (check-expect (ss 1)
                        (beside (square 10 "solid" "blue")
                                (square 10 "solid" "blue")))
```

PART (B) [6 MARKS]

Explain, step-by-step, what happens when (ss 2) and (ss 3) are run.

```
Solution: For (ss 2) I replace the parameter n throughout by 2.
          since 2>0, I use the "else" question/answer pair, which
          says I put a solid blue square as wide as (ss 1) beside
          (ss 1).  From my check-expect, I know that (ss 1) produces
          two solid blue squares, each 10 dots across, so (ss 2)
          produces a solid blue square 20 dots across beside two
          solid blue squares, each 10 dots across.

          For (ss 3) I replace the parameter n through by 3.
```

```
Since 3>0, I use the "else" question/answer pair, which
says I put a solid blue square as wide as (ss 2) beside
(ss 2).  From the previous question, I know that (ss 2)
is 40 dots wide, so (ss 3) produces a solid blue square
40 dots wide beside a solid blue square 20 dots wide
beside two solid blue squares, each 10 dots across.
```


END OF SOLUTIONS