1. (20 pts) This question deals with random number generators.

(a) (8 pts) Recall that the Ruby `rand(n)` function returns a random integer between 0 and $n-1$. Using the `rand` function, show how to compute the following:

A random integer between 0 and 109, inclusive. ____________________________
A random integer between 5 and 20, inclusive. ____________________________
A random even integer between 2 and 20, inclusive. _______________________
A random string from the array `fruit` shown below: ________________________

```
fruit = { "apple", "orange", "banana", "peach", "pear" }```

(b) (3 pts) Recall the linear congruential generator formula:

$$x_{i+1} = (a \times x_i + c) \mod m$$

If $a = 2$, $c = 3$, and $m = 5$, and the seed $x_0$ is 4, what sequence does this generator produce?

(c) (1 pt) What is the period of the generator above using the constants and seed as given? ________
A simple game is played with 2 standard 6-sided die as follows. The player starts with a total of 100 points. The player rolls the pair of dice. If the sum is 3, 6, 9 or 12, then the player earns a “zonk”; otherwise, the player adds the sum of the 2 die to the total. Once the player gets 4 zonks, the game ends.

Assume the roll function is specified by the specification below:

```ruby
def roll()
  Returns a uniformly distributed random integer between 1 and 6 for a simulated die.
end
```

Complete the Ruby function below that simulates this game using the roll function where appropriate.

```ruby
def play_game()
  total = ____________
  zonks = ____________
  while _________________________________ do
    die1 = ________________
    die2 = ________________
    if ___________________________________ then
      total = total + _________________________
    else
      zonks = zonks + _________________________
    end
  end
  return total
end
```
2. (20 pts) This problem focuses on principles of concurrency.

(a) (6 pts) Consider the following sorting network (shown as a wire diagram) that sorts 6 data values.

Show how this network sorts the values [8, 3, 4, 1, 7, 5] (as shown on the inputs of the network) by filling in each empty box in the network with its value. The first two comparisons are done for you.

(b) (4 pts) For the network shown above, assume each comparison is performed in time $t$.

How long would this sort take if each comparison is done sequentially? ____________

How long would this sort take if we use concurrency to its fullest extent for this network? ____________

(c) (8 pts) Manufacturing a toy requires 50 minutes in four sub-steps in the order given below:

- Glue toy piece together (10 minutes)
- Paint toy (15 minutes)
- Send toy through drying oven (20 minutes) – only one toy can be in the drying oven at a time
- Secure toy in package with instructions (5 minutes)

How many minutes does it take to manufacture 100 toys if we make one at a time sequentially, where we do not start the next toy until the current toy is completely finished? ____________

If we used the principle of pipelining using the four stages given above, how many minutes does it take to manufacture 100 toys? ____________

How is pipelining used in computers to speed up program execution?

__________________________________________________________________________________________

__________________________________________________________________________________________

(d) (2 pts) A failure or inability to proceed due to two programs or devices both requiring a response from the other before completing an operation is known by what term? ________________
3. (20 pts) The following question deals with issues involving the Internet.

(a) (2 pts) Give a short one sentence definition of the term *protocol*.
_____________________________________________________________________________________

(b) (4 pts) Using the Transport Control Protocol (TCP), which of the following does TCP support. Answer YES or NO for each property.

Delivers an ordered stream of data even when the underlying packets are received out of order. ______

Dropped packets are not detected so streaming video can be supported. ______

A packet is acknowledged by the receiver or else the transmitter will resend the packet. ______

A duplicate packet can be detected and thrown out by the receiver. ______

(c) (4 pts) A computer is assigned the Internet Protocol (IP) address: 42.128.35.199

Using the traditional (pre-1993) IPv4 standard, is this IP address a class A address, class B address, or class C address? __________

How many unique IP addresses can be supported using IPv4? __________

(d) (2 pts) Suppose that ISP provider MegaNet blocks its users from seeing webpages of its competitor SuperSpeed. What commonly accepted Internet principle does this practice violate? __________

(e) (4 pts) The IETF (Internet Engineering Task Force) has a dataflow model of the internet that has four layers. Match each layer with its job.

____ Application  A. Handles the task of sending packets across one or more networks.

____ Transport  B. Handles the physical transfer and reception of bits.

____ Internet  C. Handles splitting messages into packets for delivery.

____ Link  D. Handles requests from the user for data on the Internet.

(f) (2 pts) For each of the following protocols, identify to which of the four layers it belongs.

UDP ____________________________  HTTP ____________________________

(g) (2 pts) Based on the principle of abstraction, if a new link layer protocol were introduced, programs at the applications layer would not have to be reprogrammed to work with this new protocol. Why?
____________________________________________________________________________________
____________________________________________________________________________________
4. (20 pts) The following question involves cryptography. For your convenience, the Vigenère table is given below.

| A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   |
| B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   |
| C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   |
| D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   |
| F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   |
| G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   |
| H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   |
| J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   |
| K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   |
| L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   |
| M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   |
| N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   |
| O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   |
| P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   |
| Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   |
| S   | T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   |
| T   | U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   |
| U   | V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   |
| V   | W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   |
| W   | X   | Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   |
| Y   | Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   |
| Z   | A   | B   | C   | D   | E   | F   | G   | H   | I   | J   | K   | L   | M   | N   | O   | P   | Q   | R   | S   | T   | U   | V   | W   | X   | Y   | Z   |

(a) (4 pts) Decode the following word that was encoded using a Caesar cipher. (HINT: The first letter decodes to a vowel.)

J  U  P  X  A  R  C  Q  V

(b) (4 pts) Encode the message TURING using the Vigenère table with a key of LIST.

T  U  R  I  N  G
(c) (4 pts) Recall that strings can be treated as arrays in Ruby. For example, the following code prints out the ASCII code of each letter in the given message:

```ruby
message = "ABCDE"
for i in 0..message.length-1 do
  print message[i], " "
end
print "\n"
65 66 67 68 69
```

Complete the following function that applies a Caesar shift to a message. You may assume that `message` contains only lowercase letters and that `shift_amount` is an integer between 1 and 25, inclusive.

```ruby
def caesar(message, shift_amount)
  for i in 0..message.length-1 do
    message[i] = message[i] - 65
    message[i] = _________________________________________________
    message[i] = message[i] + 65
  end
  return message
end
```

(d) (8 pts) Alice and Bob want to communicate by encrypting messages using the RSA algorithm. Alice chooses the following values for her messages: d = 2753, e = 17, n = 3233. Suppose Bob wants to send the numerical message 2011 to Alice using RSA. Eve is trying to eavesdrop.

Which value(s) does Alice make public? ____________________________

What formula does Bob compute to create the encrypted message to send to Alice? (You do not have to compute the numerical value of this formula.) ____________________________

What formula does Alice compute to decrypt Bob’s message? (Again, you do not have to compute the numerical value of this formula.) ____________________________

If Eve gets a copy of Bob’s encrypted message, which value does she need to factor into the product of two primes in order to determine Alice’s decryption formula? ____________________________
5. (20 pts) This question deals with cellular automata, simulation, and AI.

(a) (8 pts) Complete the following Ruby function to implement a cellular automata for Rule 165. (Recall the powers of 2 are: $1, 2, 4, 8, 16, 32, 64, 128, ...$)

```ruby
def apply_rule(automaton)
  new_automaton = Array.new(automaton.length)
  for i in 0..automaton.length-1 do
    middle = automaton[i]
    if _________________________________ then
      left = 0
    else
      left = automaton[i-1]
    end
    if _________________________________ then
      right = 0
    else
      right = automaton[i+1]
    end
    if    left==1 && middle==1 && right==1 then new_automaton[i] = ______
    elsif left==1 && middle==1 && right==0 then new_automaton[i] = ______
    elsif left==1 && middle==0 && right==1 then new_automaton[i] = ______
    elsif left==1 && middle==0 && right==0 then new_automaton[i] = ______
    elsif left==0 && middle==1 && right==1 then new_automaton[i] = ______
    elsif left==0 && middle==1 && right==0 then new_automaton[i] = ______
    elsif left==0 && middle==0 && right==1 then new_automaton[i] = ______
    elsif left==0 && middle==0 && right==0 then new_automaton[i] = ______
    end
  end
  return new_automaton
end
```
(b) (8 pts) Recall the simulation for the spread of a virus in a population. In the simulation, a code of 6 represented an individual who got the virus and is no longer contagious.

Was this simulation time-stepped or event-driven? _________________________

Was this simulation grid-based or mesh-free? _________________________

Was this simulation deterministic or stochastic? _________________________

Suppose that the programmer tests for the end of the simulation using the following Ruby function:

```ruby
def done(matrix)
    for i in 0..matrix.length-1 do
        for j in 0..matrix[0].length-1 do
            return false if matrix[i][j] != 6
        end
    end
    return true
end
```

Will this always work? Why or why not? _________________________________________________
                                                                                      ____________________________________________________________________

(c) (4 pts) A simple MARS program to compute 3 * 8 is shown below.

```
y   DAT #8
x   DAT #3
acc DAT #0
mult ADD x, acc ; add x to acc
    SUB #1, y ; subtract 1 from y
    JMN mult, y ; jump to mult if y != 0
    DAT #0
end mult
```

Show how the program would be stored in the computer by filling in the missing blanks below.

HINT: MARS uses the computational principle of relative addressing.

```
0000: DAT #0 #8
0001: DAT #0 #3
0002: DAT #0 #0
0003: ADD ______, ______
0004: SUB #1 , ______
0005: JMN ______, ______
0006: DAT #0 #0
```