Question 1.

The analysis of Ethernet performance in the paper by Metcalfe and Boggs makes the simplifying assumption that each computer will transmit during a given slot with probability $\frac{1}{4}$, where $N$ is the number of stations trying to transmit. In reality, we know that the probability that a station will transmit depends on the number of collisions it has experienced rather than on $N$. If a computer has experienced $K$ collisions, it chooses a random delay value from the set $\{0, 1S, 2S, \ldots, (2^K - 1)S\}$. Therefore, the probability it transmits in any given slot is $\frac{1}{2^K}$.

Metcalfe and Boggs make this simplifying assumption in order to obtain a formula that approximates the expected number of slots that will be wasted either by being left idle or because a collision occurs during the slot. Any formula for the expected number of wasted slots that accurately reflects the actual dynamics of the exponential backoff algorithm would be extremely complicated. To appreciate this, we would like you to determine a formula for the expected number of slots that would be wasted in a single round of a very specific collision resolution scenario.

Suppose that two computers, A and B, attempt to transmit simultaneously on an idle Ethernet. The process of collision resolution in which they will participate can be divided into “rounds.” In round 0, both stations transmit immediately and collide. In round 1, they both randomly choose a delay from the set $\{0, S\}$ and transmit after this delay. In this round, the probability of a collision is $\frac{1}{2}$. If they collide again in round 1, then they engage in another round (round 2) in which they choose delays from the set $\{0, S, 2S, 3S\}$. The total number of slots wasted during this process will be the sum of the slots wasted in each round. It is obvious that the maximum number of slots wasted in round $K$ is $2^K$. The actual number of slots wasted in a given round, however, depends on the random delays selected.

For example, in round 1, there are four equally likely outcomes. Station A may choose delay 0 while station B chooses $S$. In this case, A will immediately transmit successfully and no slots will be wasted. Similarly, if B chooses delay 0 and A chooses $S$, no slots will be wasted. On the other hand, if both A and B choose 0, they will both transmit immediately and collide. After this collision, they will both move on to round 2 immediately, so round 1 will only waste 1 slot. On the other hand, if the both choose a delay of $S$, one slot will be wasted by being left idle and another slot will be wasted due to collision. To compute the expected number of slots wasted in round 1, we simply sum the number of slots wasted in each of the four possible outcomes time the probability of each outcome ($\frac{1}{4}$) to obtain:

$0 \times \frac{1}{4} + 0 \times \frac{1}{4} + 1 \times \frac{1}{4} + 2 \times \frac{1}{4} = \frac{3}{4}$

That is, the expected number of slots that will be wasted during round 1 will be $\frac{3}{4}$.

Assuming that A and B do collide during round 1 and move on to round 2:

a) What is the largest number of slots that can be wasted during this second round.

b) How many distinct backoff time combinations will lead to wasting exactly 2 slots during this second round.

c) Determine the expected number of slots that will be wasted during round 2. Justify your answer.

Question 2.

In their analysis of Ethernet, Metcalfe and Boggs used the variable $A$ to represent the probability that exactly one computer would attempt to transmit in a given slot. Under the assumption that each station randomly chooses to transmit in a given slot with probability $\frac{1}{Q}$, where $Q$ is the number of computers trying to transmit, they showed that

$$A = \left(1 - \frac{1}{Q}\right)^{(Q-1)}$$

Of course, every student should know their $ABC$’s, rather than just their $A$’s! So assume $B$ is the probability the slot goes idle, and $C$ is the probability that a collision occurs during a slot. That is:

$$A = \text{Prob( Exactly 1 computer transmits during a slot )}$$

$$B = \text{Prob( Exactly 0 computers transmit during a slot )}$$

\footnote{Note that in these two situations it is not reasonable to assume that the station that chooses a delay of $S$ begins a successful transmission in slot 1. This is because most packets take longer to transmit than a single slot time. So, if one station delays 0 and the other delays $S$, when the station that picked the longer delay finishes its delay it will typically find that the network is already busy finishing the transmission that was started with no delay and wait (persistently) until that transmission is complete. At that point, the two computers are likely to start a new set of transmission attempts with collisions and backoffs.}
\[ C = \text{Prob( More than 1 computer transmits during a slot )} \]

(a) Using the same assumptions Metcalfe and Boggs employed when deriving their formula for \( A \) in terms of \( Q \), derive a formula for \( B \) in terms of \( Q \).

(b) Derive a formula for \( C \) in terms of \( Q \). (Hint: What can you say about the sum of \( A \), \( B \), and \( C \)?)

(c) Given \( A \), Metcalfe and Boggs also derive a formula for \( W \), the expected number of slots between successful transmissions (\( W = \frac{1-A}{A} \)). Derive a formula for the expected number of consecutive idle slots that will be observed on an Ethernet using the same assumptions Metcalfe and Boggs employed when deriving their formula for \( W \).

**Question 3.** The code in Figure 1 defines a `StringCollection` class that contains some useful, but very poorly named, methods. Suppose that we have executed the statements:

```java
StringCollection words = new StringCollection(10);
words.add( "parts" );
words.add( "the" );
words.add( "all" );
words.add( "than" );
words.add( "greater" );
```

Indicate the String or int value that would be produced by each of the following expressions. Also provide a brief description of what the first method invoked in each expression does in general to a `StringCollection`.

a) \( \text{words.toString()} \)

b) \( \text{words.stem()} \)

c) \( \text{words.cubits()} \)

d) Now, suppose we execute the statement:

```java
words.deport( "all" );
```

Describe the string that would be returned by evaluating the expression

```java
words.toString()
```

after the statement that invokes `deport` and describe the general operation performed by the `deport` method.
public class StringCollection {

    private int size = 0;
    private String[] contents;

    public StringCollection(int maxSize) {
        contents = new String[maxSize];
    }

    public void add(String element) {
        if (size < contents.length) {
            contents[size] = element;
            size = size + 1;
        }
    }

    public String toString() {
        String result = "";
        for (int i = 0; i < size; i = i + 1) {
            result = contents[i] + " " + result;
        }
        return result;
    }

    public void deport(String target) {
        int i = 0;
        while (i < size - 1 && !contents[i].contains(target)) {
            i = i + 1;
        }
        if (contents[i].contains(target)) {
            contents[i] = contents[size - 1];
            size = size - 1;
        }
    }

    public String stern() {
        return contents[size - 1];
    }

    public int cubits() {
        int result = 0;
        for (int i = 0; i < size; i = i + 1) {
            result = contents[i].length() + result;
        }
        return result;
    }
}

Figure 1: Code for StringCollection class