Programming code requirements: If you write a program for this project and it is not C or C++, be sure to include all necessary files for your program to run. Include any instructions necessary for compiling/running the program. Ideally I want to be able to obtain an executable that runs with minimal interaction on my part (at most I supply the name of the input file – either command line or interactive – and number of stations(part2).

Part 1a – Required for CPE448 (6 pts) and 548 (3 pts)
This project is based off of Chapter 2 text book problem 49.
Each of five stations in a network has one packet to transmit. Determine the time that one of the stations is the first to successfully transmit (no collision). At t=0(slot 0), all 5 stations try to transmit and go into the **exponential backoff** part of the transmission protocol (one time slot is 51.2 uS, therefore, possible transmission times for a station are multiples of 51.2 uS.)

- **More information on the exponential backoff procedure for this problem is provided on the next page.**
- **Label the stations starting with 0. So your output should contain names of station 0, station 1, station 2, station 3 and station 4**

Turn in your answer in table form showing which stations transmit or are scheduled to transmit during each time slot. Slot 0 is the initial slot (so all five stations are listed) for transmissions at t=0 (slot 0). Slot 1 is the next slot and it starts at time t= 51.2 us, etc. Lastly, provide the time when the first station is able to successfully transmit and the number of that station (i.e. Station 1 transmits successfully in slot 5 which occurs at time 256 uS)

Use the input file Part1_rn.txt for the random numbers needed for this problem. This file contains more than the necessary number of values to successfully complete the problem. If you run out of numbers before finding a solution, then you are making a mistake somewhere. You can solve this problem by hand or use a computer program. If you use a program, submit the source code on Canvas and **give it the name Project3_part1A**.

Part 1b – Required for CPE448 (9 pts) and 548 (6 pts)
For this part each station has 1 packet to transmit. Determine the time when each of the stations make a successful transmission. Need to provide the same information as in part a, except that the information is required for all five stations. **Use the input file Part1_rn.txt for the random numbers needed for this problem.** . If you use a program different from part1A, submit the source code on Canvas and **give it the name Project3_part1B**.

Part 1c - Required for CPE448 (15 pts) and 548 (6 pts)
For this part, all stations have an endless supply of packets to transmit. After a station successfully transmits a packet, it skips 7 slots (i.e. if it transmits in slot 4, it skips slots 5 through 11 and tries its next packet in slot 12) before attempting to transmit its next packet. A station that successfully transmits a packet will reset its collision count to 0. At what time does the last station successfully transmit its first packet (that is give the minimum time when all stations have successfully transmitted at least one packet). **Use the input file Part1_rn.txt for the random numbers needed for this problem.** If you use a program different from part1A or part1B, submit the source code on Canvas and **give it the name Project3_part1C**.

Part 1 (a,b,c) output: For all parts of Part 1, the Information to submit(on paper) as an answer should be the slot number, the station(s) transmitting during that slot and the next slot backed off to for each station that collides. If a station successfully transmits at that time slot, output that fact and give the time of the transmission (slot number * 51.2uS). See Sample_output_part1b(c).txt

Part 2 – Required for CPE548 (15 pts), Extra Credit for CPE448 (6 pts)
Chapter 2 Text Book Problem 50 – **This part will require a program to be written.** Same analysis ***as performed in part 1a***. Only difference is that each simulation averages the transmission time for 10 runs for the number of stations specified. The number of stations to use for the three simulations is 20, 40 and 100. More information on this problem is provided on the next page. The files containing the random numbers for use with the simulations are shown below:

For 20 stations input file is: **`Part2_20_rn.txt`**
For 40 stations input file is: **`Part2_40_rn.txt`**
For 100 stations input file is: **`Part2_100_rn.txt`**

These files contain more than the necessary number of values to successfully complete the problem. If your program does not find a solution using these files, then there is a problem in your program.

The answer you submit(handed in on paper) for each simulation should be the information for the station that successfully transmits for each run and the average of the time(from all 10 runs) when the first station successfully transmits. See Summary_output_part2.txt for a sample of what to submit. Submit a copy of your code on Canvas and give it **the name Project3_part2.**

Project 3 addendum

Part1a: Determine at what time the first station successfully transmits given 5 stations try to transmit at t=0 (in slot 0). Each slot is 51.2 micro seconds, and all transmissions from all stations start at the slot boundary. Use exponential backoff as described in the book. Therefore, after the first transmission attempt, all stations collide. So they either back off to slot 1 or slot 2. If two or more stations collide in slot 1 (their second collision), they will backoff to slot 2, 3, 4, or 5.

To determine the backoff, the file Part1_rn.txt is available from Canvas. This file contains enough random numbers to determine the time of the first successful transmission (which will be a multiple of 51.2 uS)

For each slot where a collision occurs (two or more stations attempt a transmission), **order the stations from lowest number to highest number**. Then read a number from the input file starting with the lowest station number first. To determine the number of slots to backoff, modulo arithmetic is performed on the number read to determine the number of slots to backoff. The number of collisions that a station has experienced determines the number ($2^{numCollisons}$) to use in the modulo operation.

For example, suppose that a collision occurs in slot #3 and that this collision results in a station having had 2 collisions so far. With 2 collisions, a station will backoff (skip) 0, 1, 2 or 3 slots. So, if the current slot is slot #3, then this stations' next transmit time will be in slot #4 (0 for backoff), slot #5 (1 for backoff), slot #6 (2 for backoff) or slot #7 (3 for backoff). The number of slots to back off is determined by reading a number from the input file and performing modulo arithmetic with $2^{numCollisons}$. For the above conditions stated, suppose the number read from the input file for the station is 23451. With two collisions, the number of slots to backoff is found from

23451%4 which equals 3. Therefore, the station will skip the next three slots (slots 4,5 and 6) before attempting a transmission in slot #7.

For your solution, for each time slot write out which stations tried to transmit, and if two or more collide, then provide the next slot they will attempt a transmission. If no station transmits in a slot, then state "No station transmission attempted".

Part 2:
Perform 10 runs for each set of stations– use a loop and make all reads continuously from the input file (one for 20 stations, one for 40 stations and one for 100 stations) provided– average the 10 times of the first successful transmission for your answer.

For the 10 runs, the input file is opened once, and the numbers are read from it continuously until all 10 runs are complete. If run 1 uses 57 numbers from the file, run 2 starts with the 58th number, and so on.
Your successful transmit times for each run will be multiples of the 51.2 uS – i.e if transmission occurs on the 10th slot, then the time to transmit is 10*51.2 or 512 micro seconds. However, the average transmit time for the 10 runs may not be a multiple of 51.2 micro seconds.

For part2, find the *__first successful transmission__* time of any station – in other words perform the task as specified in part 1a not part 1b or part 1c.