1.) Uniform RNG:

Implementation :

The first part of the problem requires the implementation of the X^2 test. To perform this the program generates 400 random numbers using the uniform RNG. The random numbers are then split across 10 buckets between the interval 0-1. A series is then calculated based on the expected number of numbers in the bucket (400/10) and the actual number in the buckets. This value is then compared to the critical value (16.919) that depends on the alpha value (.05). If the series value is greater than the critical value the numbers generated are not likely random.

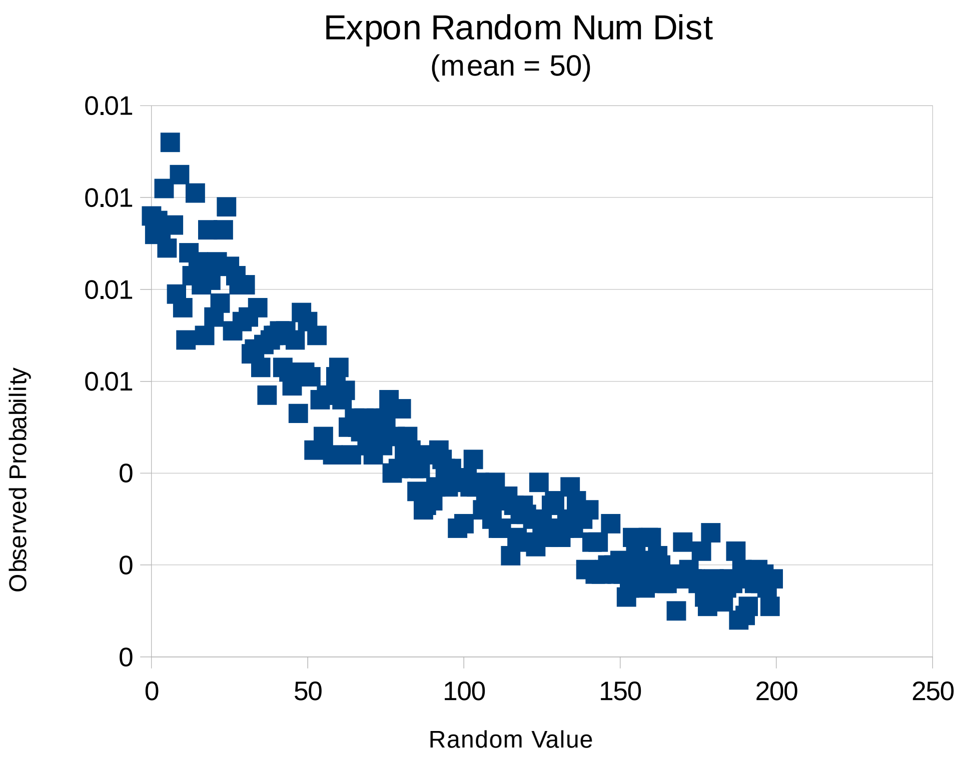
Experimentation:

The program was ran with various seeds. The program was never able to reject the hypothesis. Therefore, my confidence in the randomness of the uniform RNG is strong.

Exponential RNG:

Implementation:

This problem required calculating probability density and graphing it. Just as I did with the uniform RNG I generated 400 random numbers, divided them into buckets, and finally graphed these numbers.



2.)

Implementation:

This problem required the implementation of a M/M/1 FIFO queue simulation with a constant average service time set to 1. Average Inter-arrival times varied but for all runs P < 1. As the simulation progressed two pieces of data about the simulation were tracked: average number in system, and average wait time in system. Random numbers used for setting individual arrival and exit times were generated using an exponentially distributed random number generator and the service time mean and inter-arrival time mean as the seed.

To implement this solution the main structures used are event records, an event heap, and a queue. Event records are generated to indicate what kind of event occurs in the simulation. In my program an event can be an arrival of a customer or the exiting of a customer. Events are scheduled and placed into the heap. This way when extracting from the heap it can be guaranteed that the returned event is the most recent and therefore correct event.

When an arrival event is extracted from the heap a new arrival event is generated and also stored in the heap. If the server is not busy a exit event is also scheduled. However, if the server is busy the arrival event is placed in the queue. When an exit event is extracted from the heap the state of the server must be considered. If the queue is not empty a “customer” is extracted from the queue, an exit event is scheduled for that “customer”, and the count of served customers is incremented. If the queue is empty the server state is set to not busy and the served customer count is incremented.

Experimentation:

In order to calculate the average number of people in the system slight modifications to the program had to be made. I used an array to create buckets that correspond to number of people in the system at the current time. Every time the clock was updated in the simulation the program checks to see how many people are in the system (including customers being served). The program then looks at the clock to determine how long this number of people have been in the system. This time is then added to the corresponding bucket. Finally, each bucket valued is added together and divided by the final clock time to get the average people in the system.

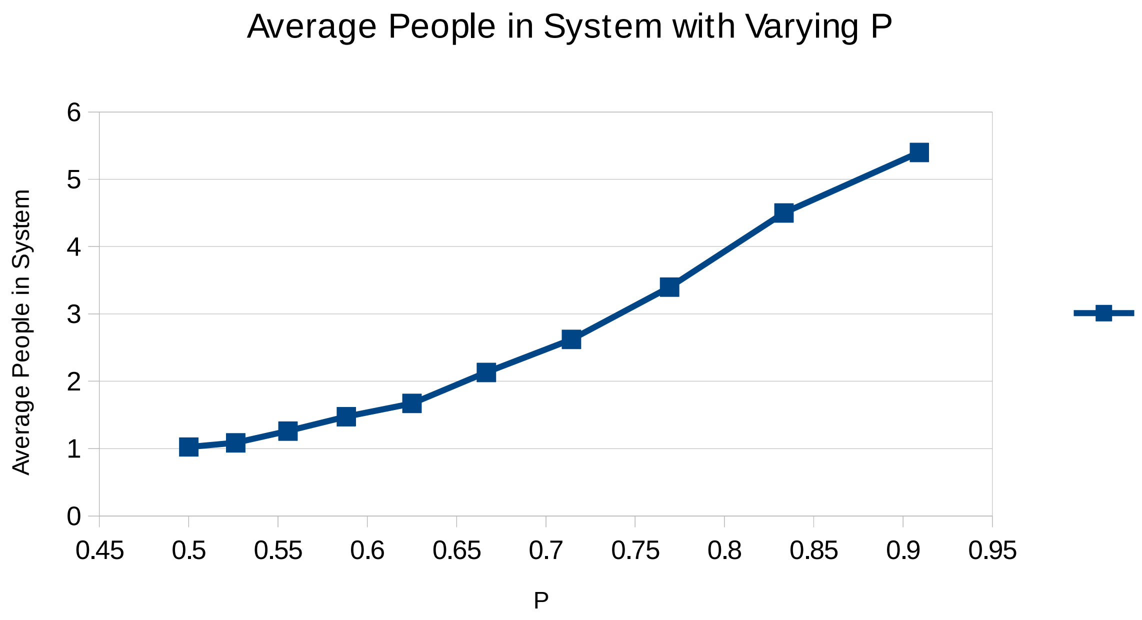
To calculate the average wait time in the system another slight modification had to be made. When a customer was extracted from the queue the program looks at the clock and calculates the time spent in the queue based by subtracting the customer's arrival time from the current time. This value of time spent waiting is added to a variable which holds the total time spent waiting for all customers. After the simulation is over this value is divided by the total number of customers and therefore indication the average wait time in the system.

During the experimentation the simulation was ran with 10 different values for P. Each value of P was ran 5 times and averaged together to get a more sound value. The simulation terminates upon the serving of 2000 customers.

Results:

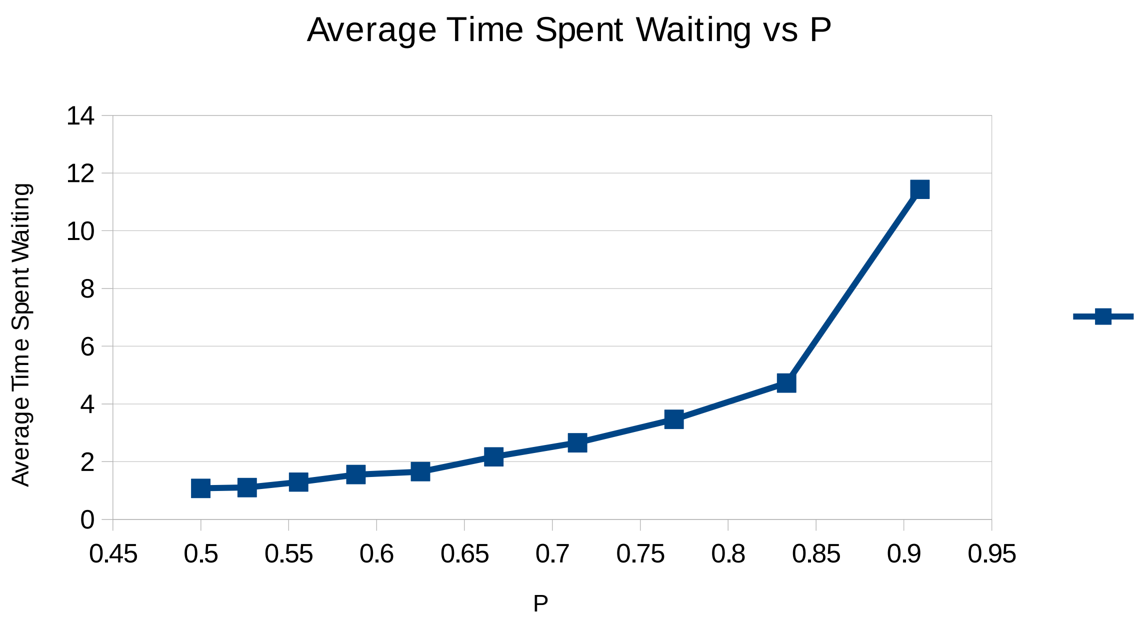
Average Number of people in system

P	Averaged Results
0.9090909090909091	5.399354
0.8333333333333334	4.498722
0.7692307692307692	3.396552
0.7142857142857143	2.620168
0.6666666666666666	2.1309180000000003
0.625	1.670044
0.5882352941176471	1.474032
0.5555555555555556	1.2600540000000002
0.5263157894736842	1.085826
0.5	1.023658



Average Time Spent Waiting

P	Averaged Results
0.9090909090909091	11.435402
0.8333333333333334	4.725282000000001
0.7692307692307692	3.4625560000000006
0.7142857142857143	2.6531960000000003
0.6666666666666666	2.166678
0.625	1.6538979999999999
0.5882352941176471	1.549628
0.5555555555555556	1.2898859999999999
0.5263157894736842	1.1019619999999999
0.5	1.073272



Discussion:

Both the average time spent waiting and average number of people in the queue gave results that were expected. As the value of P increases so does the rate at which customers are arriving. Therefore, as customers arrive more frequently the number of people who will have to wait (get queued) will increase thus increasing the average. As the size of the queue grows so does the amount of time that queued customers must wait. Therefore, the average time spent waiting increases as well.

3.)

Implementation:

This problem requires a slight modification to the program from problem 2. In this simulation the server will go on vacation (not serve any customers) for some exponentially distributed random amount of time every time there is no customer to serve. Just as in problem 2 the average service rate remains at 1 and the inter-arrival will vary. The mean used to generate random vacation time lengths is a value K * average service rate.

To implement this simulation a new event had to be created and code had to be written to handle it. The new event was an end of vacation event and uses the same event record structure as every other event. The code to handle this event had to consider the status of the queue. If the queue was non empty a customer was extracted from the queue and an new exit event is created for them. However, if the queue was empty the server would go on vacation again. Thus, a new end of vacation event must be created.

Experimentation:

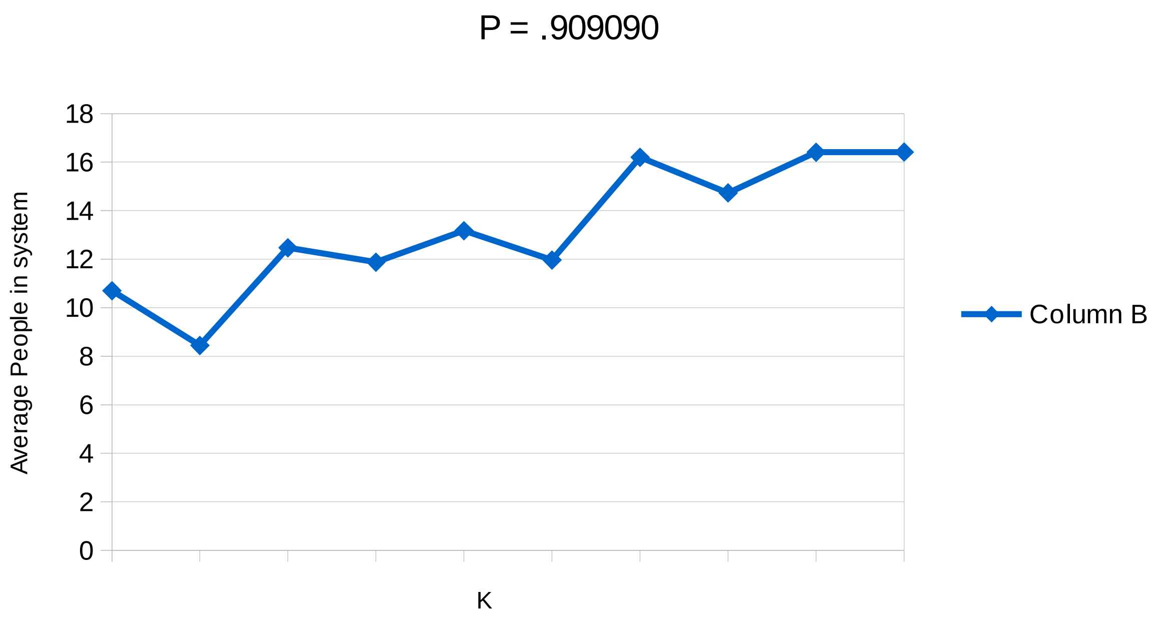
Just as in problem 2 average number of people in the system and average time spent waiting was calculated. No changes in the program had to be made to calculate the average number of people in the system. However, just like in the handling of an exit event the time customers spent waiting had to be considered for when the server was on vacation. Thus the variable that contains the time spent waiting by all customers must have the time spent in the queue by customers added to it when the server is on vacation.

The simulation was ran two different times. Once with a P value of .9090909 and another time with value .625. The value for K ranged from 1-10 for both simulations. Just like in problem 2 each simulation was ran 5 times and their values are averaged. The simulation was terminated once the 2000th customer had been served.

Results:

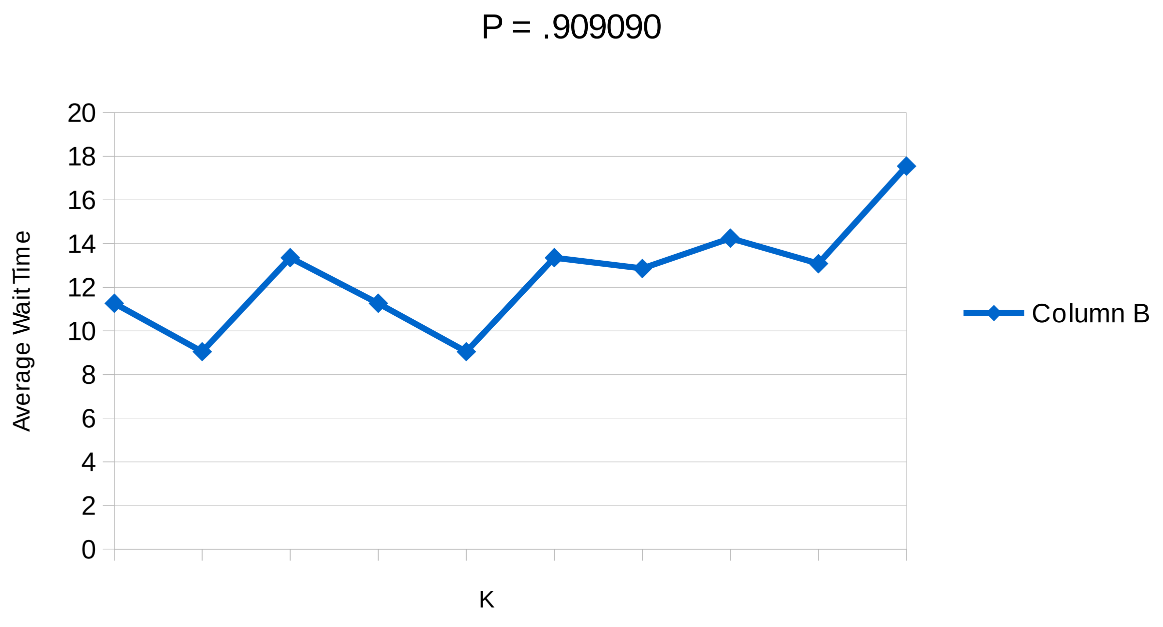
Average Number of people in system (P = 0.9090909090909091)

k	Average Value
1	10.695926
2	8.443196
3	12.469454
4	11.875726
5	13.172966
6	11.962121
7	16.195166
8	14.72929
9	16.404686
10	16.412728



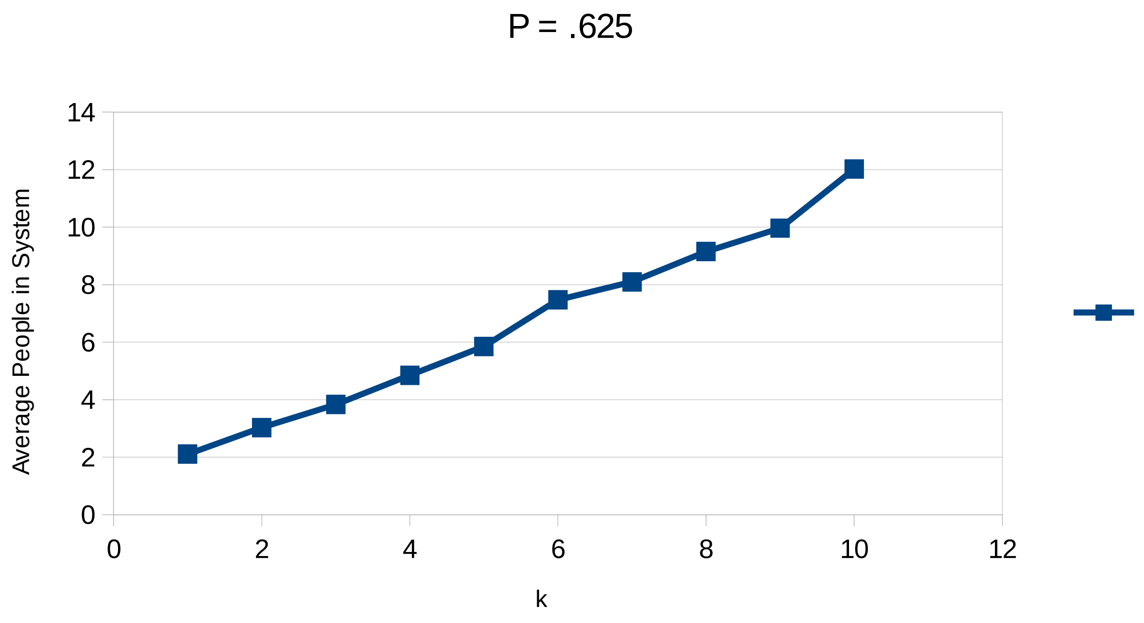
Average Wait Time in system (P = 0.90909090909091)

k	Average Value
1	11.26587
2	9.047616
3	13.354834
4	11.265868
5	9.0476161
6	13.354834
7	12.85766
8	14.247174
9	13.082756
10	17.548958



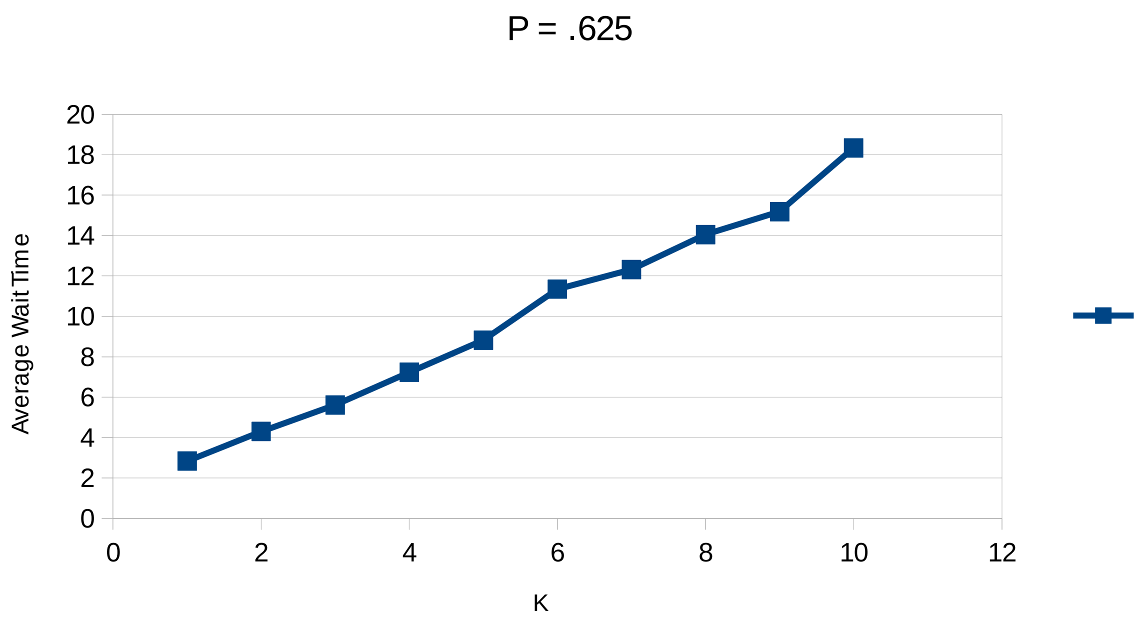
Average People in system (P = .625)

k	Average Value
1	2.107072
2	3.028774
3	3.833666
4	4.848202
5	5.851342
6	7.468226
7	8.09763
8	9.15322
9	9.964914
10	12.01957



Average Wait Time 1

k	Average Value
1	2.838926
2	4.297974
3	5.610948
4	7.232772
5	8.823004
6	11.347694
7	12.31874
8	14.045318
9	15.185414
10	18.336394



Discussion:

The results in this experiment were quite interesting. The first thing to compare is the difference between people in the system and wait times between the system that does not take vacations and the system that does. As expected the systems that take vacations have a higher number of people in the system and longer wait times on average.

It was expected that as K increased the amount spent waiting and number of people in the system would increase as well. During the experimentation with P=.90909 there was some increase in but impact was not as large as one might suspect. I believe the reason for this is that P is very close to 1. Therefore, customers are arriving almost as fast as they are being served. This means that the server will rarely be idle and will not frequently go on vacation. Therefore, the impact of the server being able to go on vacation is rare.

However, the impact of vacation when P=.625 was quite large and increased steadily. Since customers were not arriving as fast as in the previous experiment the server was able to go on vacation more frequently. This means that much of the simulation time was spent with the server on vacation. Therefore, the average size of people in the system grows, as does the average time spent waiting.