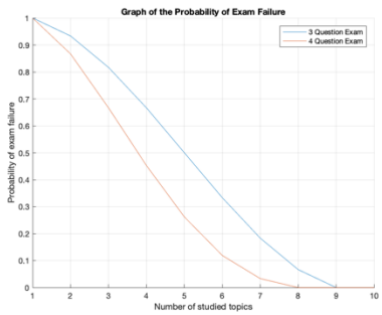
STU33009 Final Assignment 2019-20
Student name: Davy Nolan
Student number: 17330208

1(a) Since we are looking for how many unordered combinations of 3 topics that an exam can have, we can simply get a combination of 3 from the set of 10 topics: $\binom{10}{3} = 120$ different combinations of topics.

1(b) Out of the total possible combinations of topics that could appear on the exam (120), the probability that none of the n topics that I studied come up would be the number of combinations of 3 topics from the topics that I didn't study out of 120. The number of topics I didn't study is $10 - n$ so therefore the expression for this probability is: $\frac{\binom{10-n}{3}}{120}$.

1(c) To derive an expression for the probability that fewer than 2 of the n topics I studied come up in the exam, we must get the probability that 1 of my studied topics comes up in the exam and then add it to the probability that none of my studied topics comes up in the exam. There are n ways of selecting a topic that I studied to come up in the exam and there are $10 - n$ ways of choosing the next two topics. Therefore the probability that 1 of my studied topics comes up in the exam is $\frac{\binom{n}{1}\binom{10-n}{2}}{120}$ and we must add this to $\frac{\binom{10-n}{3}}{120}$ to get the final expression. The final expression is: $\frac{\binom{n}{1}\binom{10-n}{2}}{120} + \frac{\binom{10-n}{3}}{120}$.

1(d) Following the method from part (a), the total number of combinations of 4 topics from a possible 10 is $\binom{10}{4} = 210$. Using the same method as part (b), the probability that none of the n topics that I studied come up is $\frac{\binom{10-n}{4}}{210}$. Using the same method as part (c) the probability that 1 of my studied topics comes up in the exam is $\frac{\binom{n}{1}\binom{10-n}{3}}{210}$.To get the probability that I fail the exam with 4 options all worth 33% can be calculated by adding these 2 expressions as this is the same as saying the probability that less than 2 of the n topics I studied come up in the exam. The final expression is: $\frac{\binom{n}{1}\binom{10-n}{3}}{210} + \frac{\binom{10-n}{4}}{210}$.



1(e) I created the MATLAB function stochastic_sim(n) to show a stochastic simulation of the exam setup with 3 questions. This function takes n as its only input and makes use of the randsample(n,k,replacement) function to create a random set of n studied topics from 1-10 called n_topics. The function then uses randsample(n,k,replacement) with replacement set to false to create a random set of 3 topics which represent the questions on the exam called q_topics without replacement. I chose to use this function as it returns a row vector containing a random choosing of the integers from 1-n without repeating elements (i.e. without replacement). The stochastic_sim() function then loops through n_topics and q_topics, comparing their elements to see if any are the same. If it finds that 2 elements are the same from both vectors, a count called same_topics is incremented. At the end of this function, the count is checked to see if it is less than 2, if so random variable X is returned as 0 (i.e. the student failed the exam as less then 2 studied topics came up) and if greater than or equal to 2, 1 is returned (i.e. the student passed the exam as 2 or more studied topics came up). Please find function in Appendix.

1(f) I extended my stochastic_sim(n) function from part (e) so that it now takes 2 arguments: stochastic_sim(n,no_of_runs). This second argument is the value N. Instead of just returning X as either 0 or 1, it now initiates X as an array of random variables where X_i can either be 0 or 1. When X is finished being filled with values, the sum of all values of X are divided by N to get the empirical mean and this

value is returned. Outside of this function, the program uses the expression from part (c) and takes it away from 1 to get $E[X_i]$. Using the empirical mean from stochastic_sim() the standard deviation is calculated and this value is used to calculate the upper CLT and lower CLT. The results can be found in the table below.

95% CLT confidence interval, n=7, N=1000	95% CLT confidence interval, n=7, N=10000
0.7923 < Y < 0.8411	0.8090 < Y < 0.8244

1(g) Since the confidence interval from part (f) is at a 95% confidence level, the Y values generated should in theory, lie within the confidence intervals roughly 95% of the time. The more times the simulation is run, the more accurate the Y values will be, however this is a trade for program performance. For accuracy reasons I chose to run this 1000 times. For N=1000 (7.66 secs runtime), Y was within the confidence interval 95.4% of the time and for N=10000 (113.01 secs runtime), Y was within the confidence interval 95.0% of the time. In comparison, running the simulation 100 times gives the following results: for N=1000 (0.93 secs runtime), Y was within the confidence interval 94.0% of the time and for N=10000 (11.79 secs runtime), Y was within the confidence interval 93.0% of the time. Comparing these results, it would suggest that for larger sizes of N, you would require a more runs for more accuracy since for 100 runs, when N=1000, Y seems to be less affected than when N=10000. However, even at the lesser amount of 100 runs, the results are very close to 95% so there is not much of an accuracy trade-off(<2%).

1(h) For this simulation, I decided to create 2 arrays containing 10 weights (i.e. probabilities) called qP and nP which represent the probabilities of choosing topics for questions to be on the exam and for topics to be studied by the student respectively. Both of these arrays are initiated with each weight as 0.1 (i.e. 10% probability). After each iteration, the topics that come up on the exam are copied and stored in an array called last_year. This array is used in the next iteration to calculate the new weights in qP and nP. For qP, the corresponding probability of choosing the topics from last_year to be on the exam are increased from 0.1 to 0.2, making the previous year's exam topics more likely to show up this year. For nP, the corresponding probability of choosing the topics from last_year to be studied by the student are increased from 0.1 to 1. I decided to increase this to 1 as I made the students' study strategy to study all topics that came up on the exam last year. This allows the student to maximise their chances of studying a topic that will come on the exam since it is more likely for a topic from last year to come up this year. For creating an array of topics that the student studied, I used the datasample() function with n topics chosen from 1:10 with replacement set as false and using weights from nP. For creating an array of topics that are on the exam, I used the datasample() function with 3 topics chosen from 1:10 with replacement set to false and using weights from qP. I used the datasample() function as it allowed me to randomly choose topics from the options 1-10 without replacement and using weights. Each iteration, the weights are reset to 0.1 and then the process is repeated. The rest of the simulation performs the same as it did in part (f). The effect of these changes can be seen when comparing the results from the simulation without the exam changes with the results from the simulation with the exam changes. For the simulation without the changes, with n=5 and N=1000, the probability of passing the exam is 46.40% whereas for the simulation with the changes, with n=5 and N=1000, the probability of passing the exam is 68.50%. These results suggest that the changes to the exam with the students' new study strategy proves to be quite effective with roughly a 20% increase in the probability of passing the exam.

For the probability of a topic coming in the exam decreasing when it came up in last years' exam, I did the same as for the previous part, except I changed the weights of qP and nP. For qP, the corresponding probability of choosing the topics from last_year to be on the exam are decreased from 0.1 to 0.05, making the previous year's exam topics less likely to show up this year. For nP, the corresponding probability of choosing the topics from last_year to be studied by the student are decreased from 0.1 to 0. I decided to decrease this to 0 as I made the students' study strategy to avoid all topics that came up on the exam last year. This allows the student to maximise their chances of studying a topic that will come on the exam since it is less likely for a topic from last year to come up this year. The effect of these changes can be seen when comparing the results from the simulation without the exam changes with the results from the simulation with the exam changes. For the simulation without the changes, with n=5

and N=1000, the probability of passing the exam is 46.40% whereas for the simulation with the changes, with n=5 and N=1000, the probability of passing the exam is 64.80%. These results suggest that the changes to the exam with the students' new study strategy proves to be quite effective with roughly a 20% increase in the probability of passing the exam, similarly to the previous plan.

For the last part, I performed the same simulation as the one I did for the probability increasing in a topic showing up on the exam if it showed up last year, except I kept the weights for the exam questions being chosen at 0.1 for the entire simulation (i.e. not increasing to 0.2 at all) but I still increased the weights for the student choosing study topics from 0.1 to 1 if the topic appeared last year. For this simulation, with n=5 and N=1000, the probability of passing the exam is 48.10%. This result suggests that if a student is only choosing to study the same topics that came up last year plus 2 random topics, they will still have a similar probability to passing if they just chose 5 random topics, since the question topics are chosen uniformly at random.

2(a) The PMF plot for each of the exam questions can be seen below. This was done using the MATLAB histogram() function set to "Normalization" and "pdf" mode. Analysing Figure 1, the plot is right-skewed and illustrates that out of the sampled students, more scored lower than 40% than scored higher than 40% in question 1 with a large proportion (roughly 10%) of students scoring roughly 0% in the question. This suggests that the question would have been more on the difficult side as the students clearly found it challenging.

Quite the opposite can be said for Figure 2; this plot is left-skewed and illustrates that more students scored higher than 40% than scored lower than 40% in question 2 with a high proportion (roughly 20%) of students scoring almost 100% in the question. This suggests that the question would have been less challenging in comparison to question 1.

For Figure 3, this plot is almost symmetrically distributed. This illustrates that the mark that students received for this question was on average roughly 50%. This suggests that the question would have been difficult due to no students performing higher than 70%, however it was not difficult to earn enough marks to pass the question (>40%) . One could argue that question 3 is even more challenging than question 1 due to nobody scoring above 70% in question 3 but the question is easier to pass than question 1.

The strengths of using PMF to evaluate the relative difficulty of the questions: It is easy to see the mark which was scored the most by the students (e.g. In Figure 1, you can clearly see that a high amount of students scored 0% in question 1) and it is also good for comparing the distributions of the question results, allowing us to visualise what the majority of students scored in each questions.

The weaknesses of using PMF to evaluate the relative difficulty of the questions: It is difficult to add the different marks together visually (e.g. How many students scored more than 40% in Figure 1?); you can only roughly estimate by looking at the plot. Another weakness would be the fact that question difficulty is not always reflected by the results as other factors come into play such as timing (i.e. no students may have scored higher than 70% in question 3 due to there being time constraints on the day of the exam).

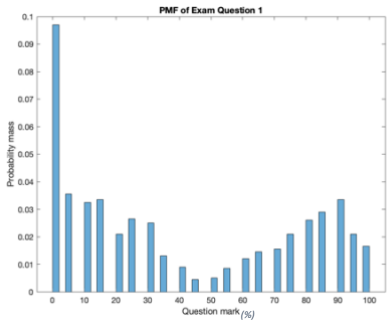


Figure 1

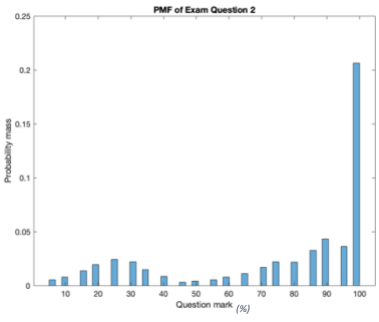


Figure 2

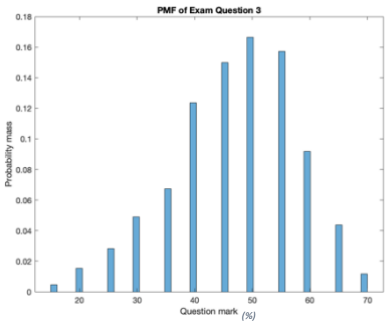


Figure 3

2(b) I decided to use the binning method for this question. I initiated a matrix which contains a 1000x2 double. I created a loop which iterates throughout all of the results to exam question 1. The current mark the loop is on is checked to see which bin it would fall into (i.e. if the mark falls into 0-10, 10-20, 20-30 etc…). My program decides which bin the mark belongs in by checking the first digit of the mark and then rounding it up and placing it in the bin with the same number (e.g. A mark of 73 will become 7 and then

round up to 8 and be placed into bin number 8). Once the mark has been given a designated bin number, the number in column 2 is incremented (i.e. Column 2 of the bins matrix contains the 10 counts for the 10 bins). This allowed me to easily track the number of marks in each bin.

I created a second loop which iterates through each mark for question 2 and then iterates through each mark for question 3. My program divides the mark from question 2/3 (depending on whether the iteration is calculating question 2 means or question 3 means) by the bin count of the bin which the corresponding question 1 mark falls into. These divisions are then stored in a matrix at the correct positions. If an estimated mean happens to fall into the same position as a previously stored value, they are added together. After all of the marks from question 2 and 3 go through this process, the matrix cond_means contains all of the estimated means for each question conditioned on the bin of question 1. The results can be seen in table 1.

For calculating the conditional variance, I created another loop similar to the loop for getting the conditional means. This loop iterates through each mark for question 2 and for question 3 and then calculates the variance conditioned on question 1 by following the formula for variance $\sigma^2 = \frac{\sum(x_i-\mu)^2}{N}$. My program takes the conditional mean (i.e. the mean of question 2/3 conditioned on question 1) from the question 2/3 mark and then squares the result and divides it by the count in the corresponding bin the question 1 mark. These results are then stored in a matrix at the correct positions. If a variance happens to fall into the same position as a previously stored value, they are added together. At the end, the matrix cond_var contains all of the estimated variances for each question conditioned on the bin of question 1. The results can be seen in the table 2.

Bins	Estimated conditional mean for Q2	Estimated conditional mean for Q3
0-10	33.5471698113208	35.1320754716982
10-20	70.0378787878788	43.4848484848485
20-30	85.7368421052631	49.1052631578947
30-40	91.0526315789474	50.3289473684211
40-50	93.5185185185185	49.4444444444445
50-60	97.5925925925926	50.5555555555556
60-70	97.9245283018868	49.8113207547170
70-80	98.7671232876713	51.8493150684931
80-90	99.6363636363635	55.1818181818182
90-100	99.9999999999997	58.3450704225352

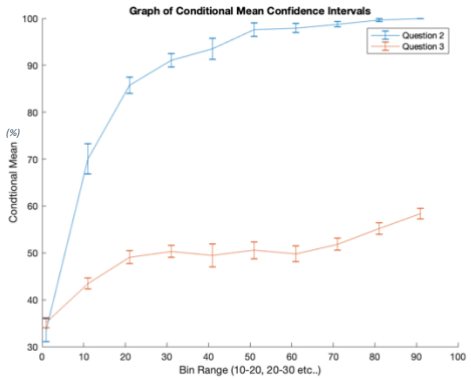
Table 1

Bins	Estimated conditional variance for Q2	Estimated conditional variance for Q3
0-10	401.945888216447	75.5485938056247
10-20	345.642504591368	47.7043158861341
20-30	72.0886426592797	43.6731301939058
30-40	38.3656509695291	30.4838988919668
40-50	34.8422496570645	41.3580246913580
50-60	13.6488340192044	20.9876543209877
60-70	12.6735493058028	38.6436454254183
70-80	5.32933008069056	30.4841433664853
80-90	2.14049586776859	43.6033057851240
90-100	1.16322273640270e-25	44.2682503471533

Table 2

2(c) For calculating the mean confidence intervals conditioned on the mark for question 1, I utilised the Central Limit Theorem (CLT) at a 95% confidence level. To calculate the CLT for each conditional mean value, I created a loop which iterated through each mean stored in the matrix cond_means. While doing this, the program accesses the previously calculated variance and square roots the value to get the standard deviation (sigma) the variance is σ^2. Usually while calculating the CLT, one would get the upper and lower CLT values, however in order to plot these intervals, since the difference between the mean and the lower CLT is the same as the difference between the upper CLT and the mean, I decided to just calculate the lower CLT. The lower CLT value is then taken away from the mean and leaves us with a value which is half the size of the error bar (i.e. confidence interval). These interval values are then stored in a matrix and later used to graph the error bars. To graph the mean confidence intervals with error bars, I used the errorbar() MATLAB function and plotted the bin ranges against the conditional mean values, including each mean's CLT confidence interval error bar. This can be seen in the graph below.

Analysing the plots,



2(d) The dataset of 1000 percentage marks for the exam questions is the training data for the linear regression. Therefore $m = 1000$ which is the amount of training examples. The input feature X is the mark for question 1 and the output feature Y is the mark for question 2 or 3 conditioned on question 1. Linear regression involves making a prediction $y = h_\theta(x) = \theta_0 + \theta_1 x$ where θ_0 and θ_1 are unknown parameters. These unknown parameters are to be chosen so that $h_\theta(x^{(i)})$ is close to $y^{(i)}$ for each of the training examples $(x^{(i)}, y^{(i)})$ where $i = 1, \dots, m$. To accurately predict a mark for question 2 and 3 based on question 1, $\theta_0, \theta_1, x^{(i)}$ and $y^{(i)}$ must reflect the estimated conditional means and variances from parts (b) and (c). Since the conditional means calculated in part (b) are not linear, one cannot assume that the mean is a linear function of the features in linear regression. To show that the linear regression model does not fit the training data, noise (i.e. random part) can be added to the prediction function: $h_\theta(x) = \theta_0 + \theta_1 x + \epsilon$. The randomness assumed in linear regression is appropriate for exam marks because we must take into account for random variables when a student is taking an exam. For example, the student could just be having an off-day or perhaps they are tired and finding it hard to focus, which may result in a poor mark but not accurately reflect the abilities of that student.

2(e) Since $X_{ij} = S_i - D_j$ where there are 1000 students and 3 questions, the PDF of X_{ij} conditioned on S_i and D_j is the product of all of the conditional PDFs $f_{X_{ij}|S_i|D_j}(X_{ij}|S_i|D_j)$ which is $\prod_{i=0,j=0}^{1000,3} S_i - D_j$. The log-likelihood function $F(\theta)$ is defined to be the natural logarithm of the likelihood function $L(\theta)$ (i.e.) $F(\theta) = \ln L(\theta)$. The likelihood function $L(\theta)$ for this case can be written as $L(\theta) = f_{X_{ij}|S_i|D_j}(X_{ij}|S_i|D_j|\theta)$. Therefore the expression for the log-likelihood of this dataset ca be written as $F(\theta) = \sum_{i=0,j=0}^{1000,3} ln\, f_{X_{ij}|S_i|D_j}(X_{ij}|S_i|D_j|\theta)$.

2(f) When using gradient descent to derive likelihood estimates from a log-likelihood function, a certain value of θ must be chosen to give different PDF results. Since we are looking for the maximum likelihood estimates for S_i and D_j we must choose the value of θ which would maximise the PDF. This can be done by using the value of θ when PDF = 0 and then reducing θ until we find the maximum of the PDF. The new value of θ can be found by performing the following expression $\theta - y * \frac{\partial * F(\theta)}{\partial * \theta}$ where y is small enough for the log-likelihood function of the new value of θ is less than the log-likelihood function of the old value of θ.

Appendix

```
% Question 1 (d)
n = 1:10;
y_exp3 = [];
y_exp4 = [];
for i=1 : 10
    y_exp3 = [y_exp3, exp3(i)];
    y_exp4 = [y_exp4, exp4(i)];
end

title('Graph of the Probability of Exam Failure')
xlabel('Number of studied topics');
ylabel('Probability of exam failure');
grid on ;
hold on ;
p1=plot(n, y_exp3);
p2=plot(n, y_exp4);
l1="3 Question Exam";
l2="4 Question Exam";
legend([p1,p2], [l1, l2]);
hold off;

function x = choose(n,k)
    x = factorial(n)/(factorial(k)*factorial(n-k));
end

function y1 = exp3(n)
    if n < 8
        y1 = ((choose(n,1)*choose(10-n,2))/120) + ((choose(10-n,3))/120);
    elseif n == 8
        y1 = ((choose(n,1)*choose(10-n,2))/120);
    else
        y1 = 0;
    end
end

function y2 = exp4(n)
    if n < 7
        y2 = ((choose(n,1)*choose(10-n,3))/210) + ((choose(10-n,4))/210);
    elseif n == 7
        y2 = ((choose(n,1)*choose(10-n,3))/210);
    else
        y2 = 0;
    end
end


% Question 1 (e)
n = 5;
x =  stochastic_sim(n);

function X = stochastic_sim(n)
X = 0;
n_topics = randsample(1:10,n,false);
q_topics = randsample(1:10,3,false);
same_topics = 0;

for i=1 : n
    for j=1 : length(q_topics)
```

```
        if n_topics(i) == q_topics(j)
            same_topics = same_topics+1;
        end
    end
end

if same_topics < 2
    X = 0;
else
    X = 1;
end
end


% Question 1 (g)
n = 7;
N1 = 1000;
N2 = 10000;
clt_l_1000 = 0.7923;
clt_h_1000 = 0.8411;
clt_l_10000 = 0.8090;
clt_h_10000 = 0.8244;

runs = 100;
in_conf_inter_1000 = 0;
in_conf_inter_10000 = 0;


for i=1:runs
    y = stochastic_sim_g(n, N1);

    if y > clt_l_1000 && y < clt_h_1000
        in_conf_inter_1000 = in_conf_inter_1000 + 1;
    end
end
for i=1:runs
    y = stochastic_sim_g(n, N2);

    if y > clt_l_10000 && y < clt_h_10000
        in_conf_inter_10000 = in_conf_inter_10000 + 1;
    end
end

result_1000 = in_conf_inter_1000/runs;
result_10000 = in_conf_inter_10000/runs;


function Y = stochastic_sim_g(n, no_of_runs)
    X = [];
    for N=1 : no_of_runs
        n_topics = randperm(10,n);
        q_topics = randperm(10,3);
        same_topics = 0;
        for i=1 : n
            for j=1 : length(q_topics)
                if n_topics(i) == q_topics(j)
                    same_topics = same_topics+1;
                end
            end
```

```
        end

        if same_topics < 2
            X = [X, 0];
        else
            X = [X, 1];
        end
    end
    Y = sum(X)/N;
end


function x = choose(n,k)
    x = factorial(n)/(factorial(k)*factorial(n-k));
end

function y1 = exp3(n)
    if n < 8
        y1 = ((choose(n,1)*choose(10-n,2))/120) + ((choose(10-n,3))/120);
    elseif n == 8
        y1 = ((choose(n,1)*choose(10-n,2))/120);
    else
        y1 = 0;
    end
end

% Question 1 (h)
n = 5;
N = 1000;
topics = 1:10;
y =  stochastic_sim_h(n,N);

function Y = stochastic_sim_h(n, N)
    X = [];
    qP = [0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1];
    nP = [0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1];
    last_year = [];
    for run_count=1 : N
        for i=1 : length(last_year)
            qP(last_year(i)) = 0.2;
            nP(last_year(i)) = 1;
        end
        n_topics = datasample(1:10, n, 'Replace', false, 'Weights', nP);
        q_topics = datasample(1:10, 3, 'Replace', false, 'Weights', qP);
        last_year = q_topics;
        qP = [0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1];
        nP = [0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1];
        same_topics = 0;
        for i=1 : n
            for j=1 : length(q_topics)
                if n_topics(i) == q_topics(j)
                    same_topics = same_topics+1;
                end
            end
        end

        if same_topics < 2
            X = [X, 0];
        else
            X = [X, 1];
```

```
        end
    end
    Y = sum(X)/N;
end

%Question 2 (a)
data = readtable('data.txt');

q1 = data.Var1;
q2 = data.Var2;
q3 = data.Var3;

figure
hist_pmf(q1);
title("PMF of Exam Question 1");
xlabel('Question mark');
ylabel('Probability mass');

figure
hist_pmf(q2);
title("PMF of Exam Question 2");
xlabel('Question mark');
ylabel('Probability mass');

figure
hist_pmf(q3);
title("PMF of Exam Question 3");
xlabel('Question mark');
ylabel('Probability mass');

function hist_pmf(u_data)
    histogram(u_data, 50,'Normalization', 'pdf');
end


%parts (b) and (c) are performed in the one script
%Question 2 (b)
data = readtable('data.txt');
data(:,4) = [];
data = table2array(data);
bins = zeros(1000,2);
cond_means = zeros(10, 2);
cond_var = zeros(10, 2);
N = 1000;
x = 100;
d = 10;
r = mod(x, d);
y = (x - r) / d;

for i = 1:N
    mark = data(i, 1);
    if (mark < 10)  %Handling bin 0-10
        bins(i, 1) = 1;
        bins(1, 2) = (bins(1, 2) + 1);
    elseif mark == 100  %Handling bin 90-100 for when the mark is 100
        bins(i, 1) = 10;
        bins(10, 2) = (bins(10, 2) + 1);
    else  %Handling bins 10-20, 20-30, 30-40, 40-50, 50-60, 60-70, 70-80,
80-90 and 90-100 for marks that are not 100
```

```
        r = mod(mark, 10);  % r is the second digit of the mark (e.g) if
mark is 73 then r is 3
        bin_no = ((mark-r)/10) + 1; % taking the digit away from the mark
and then dividing it by 10 leaves just the first digit (i.e) the bin number
        bins(i, 1) = bin_no;
        bins(bin_no, 2) = bins(bin_no, 2) + 1;
    end
end

for i = 1:2
    for j = 1:N
        %q1_cond = data(j,1); %for debugging
        q1_bin = bins(j, 1);    % The bin which question 1 is in
        mark = data(j,i+1);     % mark for question 2/3 (depending on i)
        bin_amount = bins(q1_bin,2);     % bin count for q1_bin
        result = mark/bin_amount;   % Divided the mark by the bin count for
calculating the mean
        cond_means(q1_bin,i) = result + cond_means(q1_bin,i); % Adds all
the mark/bin_amount together to get the mean for that bin
    end
end

disp(cond_means);

for i = 1:2
    for j = 1:N
        %q1_cond = data(j,1); %for debugging
        q1_bin = bins(j, 1);    % The bin which question 1 is in
        mark = data(j,i+1);     % mark for question 2/3 (depending on i)
        bin_amount = bins(q1_bin,2);     % bin count for q1_bin
        mark_mean = cond_means(q1_bin, i);
        variance = ((mark-mark_mean)^2/bin_amount);    % Divided the mark by
the bin count for calculating the mean
        cond_var(q1_bin, i) = variance + cond_var(q1_bin,i);

    end
end

disp(cond_var);

%Question 2 (c)
clt_lower = zeros(10,2);
clt_error_inter = zeros(10,2);
for i = 1:2
    for j=1:10
        mean = cond_means(j,i);
        variance = cond_var(j,i);
        n = bins(j,2);
        std_dev = sqrt(variance);

        clt_u = mean + (2 * (std_dev / sqrt(n)));
        clt_l = mean - (2 * (std_dev / sqrt(n)));

        clt_error_inter(j,i) = mean-clt_l;
    end
end


%hold on;
```

```
%plot(cond_means(:,1), 10:10:100);
hold on;
p1 = errorbar(1:10:100, cond_means(:,1),  clt_error_inter(:,1));
p2 = errorbar(1:10:100, cond_means(:,2),  clt_error_inter(:,2));
title("Graph of Conditional Mean Confidence Intervals");
xlabel('Bin Range (10-20, 20-30 etc..)');
ylabel('Condtional Mean');
l1="Question 2";
l2="Question 3";
legend([p1,p2], [l1, l2]);
```