Polytechnic Tutoring Center

Midterm 2 REVIEW – CS1133, Spring 2021

Disclaimer: This mock exam is only for practice. It was made by tutors in the Polytechnic Tutoring Center and is not representative of the actual exam given by the Academic Department.

Question 1

```
clear; clc;
nCards = 52;
nQueens = 4;
nFace = 12;
nDraws = 2; % number of times a card is drawn

nTrials = 1e5; % set some large number of trials for the simulation
DRAWS = randi([1,nCards],nTrials,nDraws);

% In the code below the first 4 cards are considered to be queens,
% and the first 12 cards are considered to be face cards.
% The four queens are included with 12 face cards, as they should be
% since queens are face cards.
firstIsAQ = DRAWS(:,1) <= nQueens;
secIsFace = DRAWS(:,2) <= nFace;

Condition = firstIsAQ & secIsFace;
nOccur = nnz(Condition); % the number of times both conditions
% (the first card being a queen and the second being a face card)
% are fulfilled
theProb = nOccur/nTrials;
disp(['The probability is: ' num2str(theProb)]);
```

Question 2

```
clear; clc;
% To generate matrix (not necessary in for your code since GRADES is given;
% Just copy and paste it into your code.
% nTests = 7; nStuds = 5; maxGrade = 100; minGrade = 0;
% GRADES = randi([minGrade maxGrade], nStuds, nTests);
GRADES = [ 61    18    16     6     9    66    45
           62    24    98    68    82    52    43
           86    89    71     4    82    98    83
           81     2    50     7    72    65     8
           58    49    47    52    15    80    13 ];
disp('The original grades are: '); disp(GRADES);
% The grades for each student are in a single row of the matrix
% (so the number of students equals the number of rows).
% Each exam is in a separate column (so the number of exams equals
% the number of columns).
```

```
[nStuds,nTests] = size(GRADES);
% Scores below 20 are counted; the student should not have more
% than 3 test scores below 20
tooLow = 20; maxTooLow = 3;

% Each iteration of the loop will go through a single row
% of the GRADES matrix, representing the exam grades for
% a single student (student i)
for i = 1:nStuds
    % Initialize variables to be used within the while loop below
    theMin = inf; % theMin will be the lowest test grade for a single
    % student; we initialize to infinity because there is no number
    % greater than infinity; we will check the grades one by one
    % in the while loop and use a logical comparison to determine
    % which is the lowest
    theS = 0; % theS will be the sum of the exam grades
    % greater than 20; we will use this to take the average of the
    % exam grades
    tC = 1; % tC will indicate the position of the exam grade
    % being considered during an iteration of the while loop.
    % We initialize the counter to 1 because we want to start
    % from the first exam grade
    nTL = 0; % we will use nTL to count the number of exam
    % grades on which student i scored below 20. We initialize
    % to zero because we have not checked any of the exam
    % grades yet, so we don't know if any exam grades are
    % below 20
    notTooManyBad = true; % notTooManyBad should be true when
    % nTL is less than 3 and false when nTL is greater than or
    % equal to 3. We initialize the variable to true because
    % so the while loop can run
    forLen = true; % this variable will check to ensure that the counter
    % tC is less than the total number of exams. Without this condition,
    % the while loop would run endlessly for students who had fewer than 3
    % exam grades under 20 (because we need a condition to be false for the
    % while loop to stop running.)

    StudentGrades = GRADES(i,:); % this is a row vector containing
    % all the exam grades for student i
    disp(['Information for student ' num2str(i) ': ']);
    while notTooManyBad && forLen
        theGrade = StudentGrades(tC); % grade of exam tC for student i
        if theGrade < theMin
            % If the grade for a given exam is less than what was
            % previously recorded for the lowest exam grade
            theMin = theGrade; % replace the value recorded for the lowest
            % grade with the current grade since it is lower
            poS = tC; % set the position of the lowest exam grade equal
            % to the position of the exam grade being considered
            % Remember, this part of the code only runs if the
            % condition theGrade < theMin is true
        end
        if theGrade < tooLow
            % If the grade is less than 20
            nTL = nTL+1; % add it to the count for nTL
            if nTL >= maxTooLow
                % If the student has reached three bad grades
```

```
                notTooManyBad = false;
                disp('There were too many bad grades.');
            end
        else
            % If the exam grade is greater than or equal to 20
            theS = theS+theGrade; % add the exam grade to the sum
        end
        tC = tC+1; % increase the counter so the next exam grade is
        % considered in the next iteration of the loop
        forLen = (tC<=nTests); % check that the counter does not
        % exceed the total number of exams
    end

    if notTooManyBad
        % If student i did not have 3 or more grades below 20
        nCount = nTests-nTL; % the number of exam grades greater
        % than or equal to 20
        theAv = ceil(theS/nCount); % average of exam grades above 20
        % for student i
        disp(['The lowest grade is ' num2str(theMin) '. It was for exam '...
            num2str(poS) '.']);
        disp(['It should be replaced with ' num2str(theAv) '.']);
        GRADES(i,poS) = theAv; % replace the lowest exam grade
        % with theAv in the GRADES matrix
    end
    disp(' '); % (this just makes it look nicer, puts an extra line
    % between the information for each student)
end

disp('The new grades are: '); disp(GRADES);
```

Extra Practice

```
clear; clc;
nTrials = 1e12; % some large number of trials (Monte Carlo simulation)
% given values:
pIN = .6; % probability that the player makes the shot
maxConMiss = 3; % the maximum consecutive misses before player gives up
atLeast10 = 10; % number of shots to see if they take before stopping

% initialize values to be used in the for loop
stoPPed = 0; % how many times player stops before shooting more than 10 times
totShots = 0; % total number of shots player has taken (in all trials)
for n = 1:nTrials
    keepGo = true; % condition to define whether while loop keeps running
    nShots = 0; % the number of shots player takes in a single trial
    nConMiss = 0; % initialize number of consecutive misses to zero
    while (nShots<=(atLeast10)) && keepGo
        theP = rand; % represents making or missing a shot
        if theP>pIN
            % the person misses the shot
            nConMiss = nConMiss+1; % increase number of consecutive misses by 1
            keepGo = (nConMiss<maxConMiss); % loop should continue running if
            % number of consecutive misses is less than 3
            if ~keepGo
```

```
                stoPPed = stoPPed+1; they stop
            end
        else
            % the person makes the shot
            nConMiss = 0; % reset number of consecutive misses to 0
        end
        nShots = nShots+1; % increase number of shots taken in that trail
    end
    totShots = totShots+nShots; % total number of shots taken
end
pMoreThan10 = 1-(stoPPed/nTrials); % probability they took more than 10 shots
before stopping is 1 minus the probability they stopped before taken more than
10 shots
avgNumShots = totShots/nTrials; % average number of shots taken
% display statements:
disp(['The player took an average of ' num2str(avgNumShots) ' shots.']);
disp(['The probability that the player will take more than 10 ...
    shots before stopping is ' num2str(pMoreThan10) '.']);
```