Polytechnic Tutoring Center

Midterm 2 REVIEW – CS1133, Spring 2020

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;
% given matrix (you can just write "MTX = given" for the exam)
MTX = [ -40    79    31    53     5    -9
         64    36   -72    53    78    54
        -74     2   -67    38    26   -65
          0    -3     5   -54    50   -57
        -51   -69   -63    28    -5   -51];
[nRows, mCols] = size(MTX); % size is only vectorized operation allowed
evenCount = 0; % initialize count of even numbers to zero
for i = 1:nRows
    for j = 1:mCols
        % one of many ways to see if a value is even:
        isEven = ceil(MTX(i,j)/2) == MTX(i,j)/2;
        if isEven
            evenCount = evenCount+1; % increase count of even numbers
            MTX(i,j) = -MTX(i,j); % change the value
        else
            % if statement to see if the odd number is positive
            if MTX(i,j)>0
                MTX(i,j) = 3*sqrt(MTX(i,j)); % change positive odd numbers
            end
        end
    end
end
toT = nRows*mCols; % total number of elements in the matrix
oddCount = toT-evenCount; % # of odds is total number minus # of evens
```

Question 2

```
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 stop will take more than 10 ...
    shots before stopping is ' num2str(pMoreThan10) '.']);
```

Question 3

```
clear; clc;
% given information
nCards = 52; % cards in a deck
maxVal = 13; % maximum value for cards
meetSum = 283; % value to meet or exceed before stopping
critAscend = 4; % number of consecutive ascending cards before stopping
nEachCard = nCards/maxVal; % how many of each card is in the deck

Deck = repmat(1:maxVal,1,nEachCard); % row vector representing the deck of cards

% initialize variables to be used in the while loop
theS = 0; % initialize the sum to zero
newLen = nCards; % number of cards in deck; will change as cards are played
nAscend = 1; % initialize number of consecutive ascending cards to 1
theVal = 0; % initialize value of drawn card to zero
asCend = true;
% loop should run while both conditions are true (if one is false, game ends)
while theS<meetSum && asCend
    preVal = theVal; % store the previous card (this is why we theVal is
    % initialized to 0; it will be used to determine whether cards are ascending
    choicE = randi([1,newLen],1,1); % the index of the card to be drawn
    theVal = Deck(choicE); % draw a card to put in the pile
    Deck(choicE) = []; % remove the card from Deck once it has been played
    theS = theS + theVal; % add value of card to existing sum of cards in pile
    % see if a card is one greater than the previous card
```

```
    oneMore = theVal == (preVal+1);
    if oneMore
        nAscend = nAscend+1; % increase the count of ascending cards
    else
        nAscend = 1; % reset the count of ascending cards
    end
    newLen = newLen - 1; % length of Deck decreases by 1 when a card is drawn
    asCend = nAscend<critAscend;
end
if ~asCend
    % if the while loop stopped because asCend stopped being true
    disp(['The game is over, you played ' num2str(critAscend) ' ascending ...
        cards in a row.']);
else
    % if the while loop stopped because the sum met or exceeded 283
    disp(['The game is over, the sum of the cards in the pile was ' ...
        num2str(theS) '.']);
end
ncardsPlayed = nCards-newLen;
disp(['You played ' num2str(ncardsPlayed) ' cards.']);
```