```
%Ahmad Malik
%ECE302-1
%Keene
%Project1

clear; clc; close all;

%% Question1

%a)
N = 1000000;  %Number of simulations

randRoll = randi(6, 3, N); % generates a 3xN matrix of random roles↙
of a dice
abilityScore = sum(randRoll, 1); %sum the columns to get the ability↙
score
score18 = length(find(abilityScore == 18));  %find the number of↙
ability scores that reach 18
fprintf("The probability of an ability score of 18 is about: %f\n",↙
score18/N);

%b)

%Generating three, (3xN) random matricies and summing/collapsing↙
them into
%single row vectors (1xN) that find the ability scores.
roll_1 = randi(6, 3, N);
abilityScore_1 = sum(roll_1, 1);
roll_2 = randi(6, 3, N);
abilityScore_2 = sum(roll_2, 1);
roll_3 = randi(6, 3, N);
abilityScore_3 = sum(roll_3, 1);

%concatenating the three ability score matricies
ability_score_matrix = [abilityScore_1',abilityScore_2',↙
abilityScore_3']';

%finding which columns have atleast one 18 ability roll and summing↙
```

```
the ammount
funScores = length(find(ability_score_matrix == 18));
fprintf("The probability of a fun score of 18 is about: %f\n",↙
funScores/N);

%c)

%using same matrix generated from part b, ability_score_matrix;
Funsum = sum(ability_score_matrix, 1);    % sum all the ability↙
scores
NumberofFreds = length(find(Funsum == 54));    % Only a Fontaine↙
would have a sum of 54 for all three ability scores.
fprintf("The probability a person would have all 18s as ability↙
scores is about: %f\n", NumberofFreds/N);

%d)

%using same matrix generated from part b, ability_score_matrix;
findCol = find(all(~diff(ability_score_matrix)));    %find all↙
columns that have repeating ability scores
findNum =  (ability_score_matrix(1,findCol));     %locate the ability↙
score numbers that repeat throughout their columns.
Find9 = length(find(findNum ==9));     %find the number of times that↙
9 is the ability score that is repeating
fprintf("The probability that you get all 9s as ability scores is↙
about: %f\n", Find9/N);


%% Question 2

%a)

%generating random HP of Trolls
Troll_HP = randi(4 , 1, N);
Avg_Troll = sum(Troll_HP)/N;%finding the expectation
fprintf("The Average HP of a Troll is about: %f\n", Avg_Troll);

%generating random damage of Fireballs
```

```
Fire_Power_Rolls = randi(2 , 2, N);
Fire_Power = sum(Fire_Power_Rolls,1);
Avg_Fire_Damage = sum(Fire_Power)/N;  %finding the expectation
fprintf("The Average amount of FireBall damage is about: %f\n",↙
Avg_Fire_Damage);

Fireball3 = 0;  %Checking he amount of times Fireball has a damage↙
greater than 3
for i = 1:N
    if Fire_Power(i) > 3
        Fireball3 = Fireball3 + 1;
    end
end
fprintf("The Probability a Firball does damage greater than 3 is↙
about: %f\n", Fireball3/N);



%b

%Sample space of Fireball damage
Fireball = [2,3,4];
%Computing pmf of Fireball damage
pmf_Fireball = zeros(1,3);
for i = 1:3
    Temp = find(Fire_Power == (i+1));
    pmf_Fireball(i) = length(Temp)/N;
end
%pmf plot of Fireball Damage
figure;
stem(Fireball, pmf_Fireball);
title("PMF of Fireball damage");
xlabel("Damage");
ylabel("Probability");
xticks(2:4);
xlim([1,5]);
ylim([0,0.75]);
```

```
%Sample space of Troll HP
Troll = [1, 2, 3, 4];
%pmf of Troll HP
pmf_Troll = zeros(1,4);
for i = 1:4
    temp = find(Troll_HP == i);
    pmf_Troll(i) = length(Temp)/N;
end
%pmf plot of Troll HP
figure;
stem(Troll, pmf_Troll);
title("PMF of Troll HP")
xlabel("HP");
ylabel("Probability");
xticks(1:4);
xlim([0,5]);
ylim([0,0.5]);

%code for c and d)

Survived = 0; %counter for amount of times Keene survives
LastSurvived = 0; %number of times a single last Troll has survived
totalHP = 0;   % Total HPs of Trolls that survived last
for i = 1:N
    Temp = 1;
    kills = 0;
    sixTrolls = randi(4, 1, 6);
    for j = 1:6
        if  sixTrolls(j)<= Fire_Power(i)

            kills = kills + 1;
            Temp = Temp * 1;    %This ensures that Temp will always↙
be 1 given that Keen slays all Trolls in order.

        else
            if kills == 5 && Temp == 1 % This statement is only↙
possible if the first 5 were killed and the last one survives
                LastSurvived = LastSurvived +1;
                totalHP = totalHP + sixTrolls(6)- Fire_Power(i); %↙
counts the net HP of the last remaining Troll after the fight
                Temp = 0;
            else
                Temp = 0;
            end
        end
    end

    if kills == 6 && Temp == 1 %This statement is only possible if↙
all Trolls are dead
        Survived = Survived + 1;
    end


end


%c
prob = Survived / N;
fprintf("Probability that Keene would survive all 6 Trolls: %f\n",↙
prob);

%d
fprintf("Expected HP value of last Troll after the fight is: %f\n",↙
totalHP/LastSurvived);


% e)

%Roll for Sword of Tuition
sword_roll = randi(6, 2, N);
Sword = sum(sword_roll , 1);

%Roll for Hammer of Tenure Denial
Hammer = randi(4, 1, N);

damage = zeros(1,N);
```

```
for i = 1:N

    if(randi(20) >= 11)
        damage(i) = damage(i) + Sword(i); %damage from sword if >=11↙
is rolled
            if(randi(20) >= 11)
                damage(i) = damage(i) + Hammer(i);% damage from↙
Hammer if >=11 is rolled again
            end
    end
end

fprintf("Shedjam's average damage: %f\n", mean(damage));
```