```
clear; clc; close all;
```

Question 1

```
% number of simulations
N = 1000000;

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% %

% a)

% roll for ability score
threeD6 = randi([1,6], 3, N);
roll3D6 = sum(threeD6, 1);

% probability that any one roll of 3 dice will generate an ability
 score of
% 18
pPerfect18 = length(find(roll3D6 == 18))/N;
disp("P[18 in one roll] = " + pPerfect18 + ", which is very close to
 theorectical probability of 0.0046");

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% %

% b) fun method to get 18

% roll for fun method
% generate a 3 x 3 x N vector
%   (3 rolls for 3d6, repeat for N trails)
threeD6 = randi([1,6], 3, 3, N);

% initialize
fun3D6 = zeros(1,N);
% sum(threeD6, 2) is the result of each 3d6 roll stored in a 3 x 1 x N
% vector
% keep the max out of the three rolls for each trial
fun3D6(:) = max(sum(threeD6, 2));

% probability of getting 18 using fun method
pFun18 = length(find(fun3D6 == 18))/N;
disp("P[18 fun roll] = " + pFun18 + ", which is very close to
 theorectical probability of 0.0138");

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% %

% c) fun method to get fred fontaine

% counter for getting a Fred using fun method
nFred = 0;
```

```
% N x 6 vector to store all 6 ability power points for all N
 characters
% generated during simulation
generatedFred = zeros(N,6);
for i = 1:N
    % same approach as part b) -- fun method
    threeD6 = randi([1,6], 3, 3, 6);
    funFred = zeros(1,6);
    funFred(:) = max(sum(threeD6, 2));
    generatedFred(i,:) = funFred;

    % check if all 6 ability powers are maxed out at 18
    if sum(funFred, 1) == 6*18
        nFred = nFred + 1;
    end
end

pFunFred = nFred/N;
disp("P[Fred fun roll] = " + pFunFred);
disp("    This turns out to be zero because the theoretical
 probability is extremely small,");
disp("    being 6.9e-12.");
disp("    Doing simulation to converge to  expected result will take a
 HUGE number of iterations,");
disp("    in the magnitude of 10e13, which requires more than 200TB of
 memory.");

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% %

% d) fun method get keene

% counter for getting a Fred using fun method
nKeene = 0;

% N x 6 vector to store all 6 ability power points for all N
 characters
% generated during simulation
generatedKeene = zeros(N,6);
for i = 1:N
    % same approach as part b) -- fun method
    threeD6 = randi([1,6], 3, 3, 6);
    funKeene = zeros(1,6);
    funKeene(:) = max(sum(threeD6, 2));
    generatedKeene(i,:) = funKeene;

    % check if all 6 ability powers are 9

    % counter for number of ability point that equals to 9
    keeneChecker = 0;

    % check all 6 ability points for the generated character
    % (if they equals to 9)
```

```
    for j = 1:6
       if generatedKeene(i,j) == 9
           keeneChecker = keeneChecker + 1;
       end
    end

    % if all 6 ability points are 9, a wild Keene has appeared
    if keeneChecker == 6
        nKeene = nKeene + 1;
    end
end

pFunKeene = nKeene/N;
disp("P[Keene fun roll] = " + pFunKeene);
```

P[18 in one roll] = 0.004589, which is very close to theorectical probability of 0.0046
P[18 fun roll] = 0.013909, which is very close to theorectical probability of 0.0138
P[Fred fun roll] = 0
This turns out to be zero because the theoretical probability is extremely small,
being 6.9e-12.
Doing simulation to converge to expected result will take a HUGE number of iterations,
in the magnitude of 10e13, which requires more than 200TB of memory.
P[Keene fun roll] = 0

Question 2

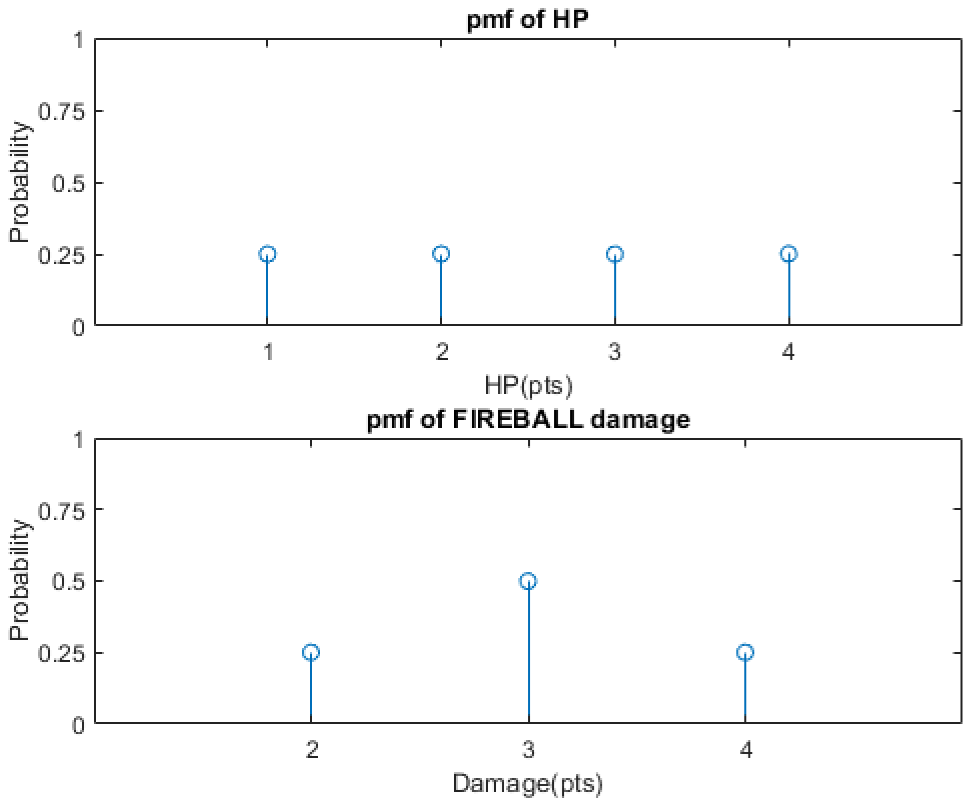
```
% number of simulations
N = 1000000;

% roll for trolls' HP
oneD4 = randi([1,4], 1, N);
HP = sum(oneD4, 1);

% roll for keene's fireball damage
twoD2 = randi([1,2], 2, N);
FIREBALL = sum(twoD2, 1);

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% %

% a)

% expectation (average) of trolls' HP
eHP = sum(HP)/N;
disp("E[HP} = " + eHP + ", which is very close to theoretical
 expectation of 2.5");

% expectation (average) of keene's fireball damage
```

```
eFIREBALL = sum(FIREBALL)/N;
disp("E[FIREBALL] = " + eFIREBALL + ", which is very close to
 theoretical expectation of 3");

% count of FIREBALLs that does greater than 3 points of damage
% (number of FB greater than 3)
nFBgt3 = 0;
for i = 1:N
    if FIREBALL(i) > 3
        nFBgt3 = nFBgt3 + 1;
    end
end

% probability that the FIREBALL does greater than 3 points of damage
pFBgt3 = nFBgt3 / N;
disp("P(FIREBALL > 3) = " + pFBgt3 + ", which is very close to
 theoretical probability of 0.25");

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% %

% b)

% sample space of HP
nHP = [1, 2, 3, 4];

% sample space of FIREBALL damage
nFIREBALL = [2,3,4];

% pmf of HP
pmf_HP = zeros(1,4);
for i = 1:4
pmf_HP(i) = length(find(HP == i))/N;
end

% pmf of FIREBALL damage
pmf_FIREBALL = zeros(1,3);
for j = 1:1:3
pmf_FIREBALL(j) = length(find(FIREBALL == (j+1)))/N;
end

% plotting
figure;

% pmf of HP plot
subplot(2,2,[1 2]);
stem(nHP, pmf_HP);
title("pmf of HP")
xlabel("HP(pts)");
ylabel("Probability");
xticks(1:4)
yticks(0:0.25:1)
xlim([0,5])
ylim([0,1])
```

```
%pmf of FIREBALL damage plot
subplot(2,2,[3 4]);
stem(nFIREBALL, pmf_FIREBALL);
title("pmf of FIREBALL damage");
xlabel("Damage(pts)");
ylabel("Probability");
xticks(2:4)
yticks(0:0.25:1)
xlim([1,5])
ylim([0,1])

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% %

% c) & d)

% generate HP for N squads of 6 trolls
squadHP = randi([1,4], 6, N);

% counter for number of times that all 6 trolls are slayed
nAced = 0;

% if there is only one troll survived,

% counter for number of times that only one troll survived
nSurvivors = 0;
% array to store the last survivor's HP
survivorHP = zeros(1, N);
% array to sotre the FIREBALL damage dealt to the last survivor
survivorFB = zeros(1, N);

for j = 1:N

    % counter for number of trolls killed in a 6-troll squad
    % reset to 0 for every iteration of simulation
    nKills = 0;
    for k = 1:6
        if squadHP(k, j) <= FIREBALL(j)
            nKills = nKills + 1;
        else
            % index of the last survivor
            iSurvivor = k;
        end
    end

    if nKills == 6
        nAced = nAced + 1;
    else
       if nKills == 5
          survivorHP(nSurvivors + 1) = squadHP(iSurvivor, j);
          survivorFB(nSurvivors + 1) = FIREBALL(j);
          nSurvivors = nSurvivors + 1;
       end
```

```
    end
end

% c)
pAced = nAced / N;
disp("P(Aced) = " + pAced + ", which is very close to theoretical
 probability of 0.343");

%d)
survivorRemainingHP = survivorHP - survivorFB;
eSurvivorRemainingHP = sum(survivorRemainingHP)/nSurvivors;
disp("E[HP of remaining troll] = " + eSurvivorRemainingHP);

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% %

% e)

% roll for Sword of Tuition
twoD6 = randi([1, 6], 2, N);
tuitionSword = sum(twoD6, 1);

% roll for Hammer of Tenure Denial
oneD4 = randi([1,4], 1, N);
tenureHammer = sum(oneD4, 1);

% initialize total damage dealt by Shedham to 0
damage = zeros(1,N);

for m = 1:N
    % if an 11 or greater is rolled on a 20-sided dice
    if(randi([1,20] ,1 ,1) >= 11)
        % Sword of Tuition attack!!!
        damage(m) = damage(m) + tuitionSword(m);
            % if an 11 or greater is rolled AGAIN on a 20-sided dice
            if(randi([1,20], 1, 1) >= 11)
                % Hammer of Tenure Denial attack!!!
                damage(m) = damage(m) + tenureHammer(m);
            end
    end
end

eDamage = mean(damage);
disp("E[Shedham's damage] = " + eDamage);
```

```
E[HP} = 2.5013, which is very close to theoretical expectation of 2.5
E[FIREBALL] = 2.9999, which is very close to theoretical expectation
 of 3
P(FIREBALL > 3) = 0.25029, which is very close to theoretical
 probability of 0.25
P(Aced) = 0.34276, which is very close to theoretical probability of
 0.343
E[HP of remaining troll] = 1.0586
E[Shedham's damage] = 4.1304
```



Published with MATLAB® R2020b