1. The best decryption is:

   ```
   WHEN I FIND MYSELF IN TIMES OF TROUBLE MOTHER MARY
   COMES TO ME SPEAKING WORDS OF WISDOM LET IT BE AND IN
   MY HOUR OF DARKNESS SHE IS STANDING RIGHT IN FRONT OF
   ME SPEAKING WORDS OF WISDOM LET IT BE LET IT BE LET IT
   BE LET IT BE LET IT BE WHISPER WORDS OF WISDOM LET IT
   BE AND WHEN THE BROKEN HEARTED PEOPLE LIVING IN THE
   WORLD AGREE THERE WILL BE AN ANSWER LET IT BE FOR
   THOUGH THEY MAY BE PARTED THERE IS STILL A CHANCE THAT
   THEY WILL SEE THERE WILL BE AN ANSWER LET IT BE LET IT
   BE LET IT BE LET IT BE LET IT BE YEAH THERE WILL BE AN
   ANSWER LET IT BE LET IT BE LET IT BE LET IT BE LET IT
   BE WHISPER WORDS OF WISDOM LET IT BE LET IT BE LET IT
   BE LET IT BE LET IT BE WHISPER WORDS OF WISDOM LET IT
   BE AND WHEN THE NIGHT IS CLOUDY THERE IS STILL A LIGHT
   THAT
   ```

   The corresponding P(f) value is -1.1138e+03;
   The inverse permutation is (5   4   6   7   1   9   2   3   8);
   We guess this is the song <*let it be*>, and the original author is *The Beatles.*

2. ‘Q’ ; ‘Z’ ; ‘J’ ; ‘X’ ; ‘K’
3. ‘E’ to ‘ ’; ‘ ’ to ‘T’; ‘H’ to ‘E’; ‘T’ to ‘H’; ‘D’ to ‘ ’
4. The code :

Part 1: Data Processing

```
% This piece of code does the job of data preprocess
% The first 842 and last 364 lines have already been deleted in the
file "War and Peace.txt"
% Following has been done:
% 1. convert all letters to upper case
% 2. Only select characters that in alph, new text is stored in
'data.txt'
clear;clc;
filename = 'War and Peace.txt';
file = fileread(filename);
upperfile = upper(file);

%by converting to ASCII, we only select characters in
alph(corresponding ASCII are 65-90 and 32)
upperfile_asc=abs(upperfile);
upperfile_asc(~(((65<=upperfile_asc)&(upperfile_asc<=90))|(upperfile_as
c==32)))=[];
upperfile=char(upperfile_asc);

%Store the selected text in data.txt
fileID = fopen("data.txt",'w');
fprintf(fileID,upperfile);
fclose(fileID);
```

Part 2: Creating log_char_freq and log_TransB Tables

```
% Run this piece of code, freq vector and TranB table will be created
and
% their transformed values (log) will be stored as log_char_freq and
log_TransB
clear;clc;
filename = 'data.txt';
file = fileread(filename);
alph =
['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R
','S','T','U','V','W','X','Y','Z',' '];
save alph

file_len = length(file);
alph_len = length(alph);

char_freq = zeros(alph_len,1);
for i=1:alph_len
    idx = strfind(file,alph(i));
    char_freq(i) = length(idx) / file_len;
end
log_char_freq = log(char_freq);
save log_char_freq

TransB = zeros(alph_len,alph_len); % Initialize TransB
for i=1:file_len-1
    char1 = file(i);
    char2 = file(i+1);
    indexOfchar1 = strfind(alph,char1);
    indexOfchar2 = strfind(alph,char2);
    TransB(indexOfchar1,indexOfchar2) =
TransB(indexOfchar1,indexOfchar2) + 1;
end

unnormalized_TransB = TransB; %Used for Question Answering
save unnormalized_TransB

Diag = diag(1./sum(TransB,2));
TransB = Diag * TransB; % Normalize each row
% Replace -Inf values in log_TransB by a small value -12
log_TransB = log(TransB);
[x,y] = find(log_TransB == -inf);
for i=1:length(x)
    log_TransB(x(i),y(i)) = -12;
end

save log_TransB
```

Part 3: Metropolis_Hasting decoder

```
% Main Part of the project: using Metropolis-Hasting algorithm to find
best
% permutation (Permu_max), apply it to the whole ciphertext for
decryption
clear;clc;
Num_MH = 20; % Run the Metropolis-Hastings algorithm 20 times
```

```
ciphertext = fileread("ciphertext.txt");
l = 9; % key length
% Initialize P_max and Permu_max
P_max = -inf;
Permu_max = [1,2,3,4,5,6,7,8,9];

load('alph.mat');
load('log_char_freq.mat');
load('log_TransB.mat');



% loop Num_MH time for searching Permutation with Maximum plausibility
value
for i=1:Num_MH
% Run Metropolis-Hastings algorithm for Num_MH times,
% each time with MaxIt iterations
% For each run of MH algorithm, f_j is randomly created for decryption
    j = randi(30,1); % Uniformly select an integer from 1 to 30
    f_j = ciphertext(j*l+1:end);% delecting the first j blocks
    [P_candidate,Permu_candidate] =
MH_decoder_fun(alph,log_char_freq,log_TransB,f_j);
    if P_candidate > P_max
        P_max = P_candidate;
        Permu_max = Permu_candidate;
    end
end

plaintext = apply_permu(ciphertext,Permu_max);% Permu_max is applied to
the whole ciphertext
P_plaintext = plausibility(alph,log_char_freq,log_TransB,plaintext);%
Calculate corresponding plausibility
% decrypted text is stored in plaintext.txt
fileID = fopen("plaintext.txt",'w');
fprintf(fileID,plaintext);
fclose(fileID);
```

functions used in the above section:
MH_decoder_fun

```
function[P_max,Permu_max] =
MH_decoder_fun(alph,log_char_freq,log_TransB,f_j)
% for given ciphertext f_j, MaxIt of interatation is runned,
% best Permutation is returned together with plausibility
    Permutation = [1,2,3,4,5,6,7,8,9]; %Initialize Permuatation as
indentity
    P_j = plausibility(alph,log_char_freq,log_TransB,f_j);
    P_max = P_j;
    Permu_max = Permutation;
    MaxIt = 6000;
    % Iterate first MaxIt/2 with slide move
    for k1=1:MaxIt/2
        New_Permu_candidate = slide_move(Permutation);
        f_star = apply_permu(f_j,New_Permu_candidate);
        P_star = plausibility(alph,log_char_freq,log_TransB,f_star);
        f_old = apply_permu(f_j,Permutation);
        P_old = plausibility(alph,log_char_freq,log_TransB,f_old);
        u = rand(1,1);
        if u < exp(min(0,P_star - P_old))
            Permutation = New_Permu_candidate; % Permutation is the one
in Markov Chain
        end

        if P_star > P_max
            P_max = P_star;
            Permu_max = New_Permu_candidate;
        end
    end

    % Iterate last MaxIt/2 with swap move
    for k2=1:MaxIt/2
        New_Permu_candidate = swap_move(Permutation);
        f_star = apply_permu(f_j,New_Permu_candidate);
        P_star = plausibility(alph,log_char_freq,log_TransB,f_star);
        f_old = apply_permu(f_j,Permutation);
        P_old = plausibility(alph,log_char_freq,log_TransB,f_old);
        u = rand(1,1);
        if u < exp(min(0,P_star - P_old))
            Permutation = New_Permu_candidate;
        end

        if P_star > P_max
            P_max = P_star;
            Permu_max = New_Permu_candidate;
        end
    end
end
```

plausibility

```
function [P] = plausibility(alph,log_char_freq,log_TransB,f)
% Compute the plausibility of given text
% Input:f where f is a text
    P = log_char_freq(strfind(alph,f(1)));
    for i=1:length(f)-1
        P = P + log_TransB(strfind(alph,f(i)), strfind(alph,f(i+1)));
    end
end
```

slide_move

```
function [permutation2] = slide_move(permutation1)
% new permutation(permutation2) is created from slide move
transformation of given
% permutation (permutation1)
% Permuation1 and Permutation2 are 1*9 vectors
    l = 9;
    b = randi(l-2,1);
    k1 = randi(l-b+1,1);
    k2 = randi([0,l-b],1);
    Deleted_Part = permutation1(k1:k1+b-1);
    Remainder_Part = permutation1([1:k1-1 k1+b:end]);
    permutation2 = [Remainder_Part(1:k2) Deleted_Part
Remainder_Part(k2+1:end)];
end
```

swap_move

```
function [permutation] = swap_move(permutation1)
% new permutation(permutation) is created from swap move transformation
of given
% permutation (permutation1)
% Permuation1 and Permutation are 1*9 vectors
    r1 = randi(9,1); r2 = randi(9,1);
    permutation = permutation1;
    a =  permutation1(r1);
    permutation(r1) = permutation1(r2);
    permutation(r2) = a;

end
```

Part 4: Question answering

```
% Q2 Five least frequent characters in <War and Peace> in increasing
order

clear; clc;
load log_char_freq; freq = log_char_freq;
load alph; alp = alph;
load unnormalized_TransB; TransB = unnormalized_TransB;
Min_inc_index = zeros(1,5); %increase order
for j = 1:5
    [m,index] = min(freq);
    Min_inc_index(j) = index;
    freq(index) = +inf;
end
Sol_Q2 = alp(Min_inc_index);


%Q3 Five most frequent transitions
Max_dec_index = zeros(2,5); %Each column represents one transition
for j = 1:5
    n = max(max(TransB));
    [row,col] = find(TransB==n);
    Max_dec_index(1,j) = row;
    Max_dec_index(2,j) = col;
    TransB(row,col) = -inf;
end
Sol_Q3 = alp(Max_dec_index);
```