```python
# Amogha Sekhar, A53301791, CSE250A
#Part A
import pandas as pd
import random
import string

#Convert to dataframe
data = pd.read_csv('hw1_word_counts_05.txt', header = None, sep= " ")
data.columns= [ 'Word', 'WordCount']
print(data['Word'])
#data.set_index('Word')

#Calculate prior probabilities P(W=w)
for i in range(6535):
    data['Prior_Probability']= data['WordCount']/sum(data['WordCount'])

#Sorting for Sanity Check
data.sort_values(by= ['Prior_Probability'], axis=0,ascending= True,inplace= True
)

#Sanity Check
data.tail(15)
data.head(14)

#************************************************************************
#Part B
#Function to predict the best letter to guess and probability
def predict(partial,exclude,remaining ):
    bad_words= set()
    for w in data['Word']:
#Check in exclude, if any word has those letters, add them to bad_words
        if any(letter in w for letter in exclude):
            bad_words.add(w)
            data[data['Word']!= w]

            continue
#Check if the positions match for words where the letters are present
    for pos in range(5):
        for index, d in data.iterrows():
            if partial[pos].isalpha() and d['Word'][pos]!= partial[pos]:
                data.drop(index, inplace= True)
            else:
                for let in partial:
                    if d['Word'][pos]!= let and let.isalpha():
                        data.drop(index, inplace= True)
```

```python
#Calculating posterior probabilities
    posterior= dict()
    for index, d in data.iterrows():
        posterior[d['Word']]= d['Prior_Probability']/sum(data['Prior_Probability'])

#Calculate most probable letter
    prob= dict()
    for let in remaining:
        prob[let] = sum((let in w) * posterior[w] for w in data['Word'])

    best_choice = max(prob, key=prob.get)
    best_choice_prob = prob[best_choice]
    return best_choice, best_choice_prob

#Generate a random word from the corpus
#word= random.choice(data['Word'])
#print(word)
#word= word.lower()

#Include keeps track of letters in the word which are guessed
#Exclude keeps track of the letters guess but not present in the word
#Remaining keeps track of the letter which are yet to be guessed

include= []
exclude=[]
remaining= list(string.ascii_lowercase)

#Take input in
include_str= input("Enter the correctly guessed letters in order: ")
exclude_str= input("Enter the incorrectly guessed letters: ")

include_str= include_str.lower()
exclude_str= exclude_str.lower()


def split(word):
    return [char for char in word]

include= split(include_str)
exclude= split(exclude_str)

def Diff(l1, l2):
    l3 = [i for i in l1 + l2 if i not in l1 or i not in l2]
    return l3
```

```
remaining= Diff(remaining, include)
remaining= Diff(remaining, exclude)
print(type(remaining))
while '-' in remaining:
    remaining.remove('-')


#Partial tries to reconstruct the word
partial= include

best_letter,best_letter_probability= predict(partial,exclude,remaining)
print('Best letter to guess:', best_letter)
print('With probability ', best_letter_probability)
    #letter= input('Enter a letter: ')
    #letter= letter.lower()

    #if letter in word:
        #print('Letter is present!')
        #partial = list(partial)
        #for i, c in enumerate(word):
            #if letter == c:
                #partial[i] = letter
        #partial = ''.join(partial)
        #print(partial)
        #include.append(letter)
        #remaining.remove(letter)
    #elif letter not in word:
        #print('Letter not present')
        #exclude.append(letter)
        #remaining.remove(letter)

#print('You guessed the word correctly!')
```