```cpp
// Code Created for CSE 250A Homework 3 Question 5
// Creator: Jingyuan Li with All rights reserved.
// 10-18-2015 At UCSD


#include <fstream>
#include <sstream>
#include <string>
#include <iostream>
#include <vector>
#include <map>
#include <cmath>

using namespace std;

vector<pair<string, double> > uniDis(vector<string> vacab, vector<int> count,
    char c){
    double sum = 0.0;
    vector<pair<string, double> > ans;
    for (int i = 0; i < count.size(); i++) sum += double(count[i]);
    for (int i = 0; i < vacab.size(); i++){
        if (vacab[i][0] == c){
            pair<string, double> p(vacab[i], count[i]/sum);
            ans.push_back(p);
        }
    }
    return ans;
}

vector<pair<string, double> > bigDis(vector<string> vacab, vector<int> count,
    string word, int budget, vector<pair<int, int> > combIndex, vector<pair<int,
    int> > lookuptable){
    int index;
    for (int i = 0; i < vacab.size(); i++){
        if (vacab[i] == word) { index = i+1; break; }
    }
    vector<pair<string, double> > ans;
    map<int, int> m;
    for (int i = 0; i < combIndex.size(); i++){
        if (combIndex[i].first == index){
            int preindex = combIndex[i].second;
            int times = lookuptable[i].second;
            m[times] = preindex;
        }
    }
    map<int, int>::reverse_iterator itr = m.rbegin();
    while (budget > 0){
        pair<string, double> p(vacab[(*itr).second-1], (*itr).first/double(count
            [index-1]));
        ans.push_back(p);
        itr++; budget--;
    }
    return ans;
}

vector<double> questionc(vector<string> sentence, vector<string> vacab, vector<
    int> count, vector<pair<int, int> > combIndex, vector<pair<int, int> >
    lookuptable){
    double lu = 1.0, lb = 1.0, sum = 0.0;
    vector<double> ans;
    vector<int> index = {2};
    for (int i = 0; i < count.size(); i++) sum += double(count[i]);
    for (int i = 0; i < sentence.size(); i++){
        index.push_back(find(vacab.begin(), vacab.end(), sentence[i]) - vacab.
            begin() + 1);
        lu *= (count[*(index.end()-1)-1]/sum);
        bool found = false;
        for (int j = 0; j < combIndex.size(); j++){
            if (combIndex[j].first == *(index.end()-2) && combIndex[j].second ==
                *(index.end()-1)){
                lb *= lookuptable[j].second/double(count[*(index.end()-2)]-1);
                    found = true; break;
            }
        }
        if (!found) { lb *= 0; }
    }
    lu = log(lu);
    lb = lb!=0 ? log(lb) : 0;
    ans.push_back(lu);
    ans.push_back(lb);
    return ans;
}

vector<double> questiond(vector<string> sentence, vector<string> vacab, vector<
    int> count, vector<pair<int, int> > combIndex, vector<pair<int, int> >
    lookuptable, vector<double> lamda){
    double sum = 0.0;
    vector<double> ans;
    for (int i = 0; i < count.size(); i++) sum += double(count[i]);
    for (int m = 0; m < lamda.size(); m++){
        double lb = 1.0;
        vector<int> index = {2};
        for (int i = 0; i < sentence.size(); i++){
            index.push_back(find(vacab.begin(), vacab.end(), sentence[i]) - vacab
                .begin() + 1);
            double lu = count[*(index.end()-1)-1]/sum, k;
            bool found = false;
            for (int j = 0; j < combIndex.size(); j++){
                if (combIndex[j].first == *(index.end()-2) && combIndex[j].second
                    == *(index.end()-1)){
                    k = lookuptable[j].second/double(count[*(index.end()-2)-1]);
                    found = true; break;
                }
            }
            if (!found) k = 0;
            lb *= lamda[m]*lu + (1-lamda[m])*k;
        }
        lb = log(lb);
        ans.push_back(lb);
    }

    return ans;
```

```cpp
}

int main(){
    ifstream myfile;
    myfile.open("vocab.txt");
    vector<string> vacab;
    string s;
    if (myfile.is_open()){
        while (!myfile.eof()){
            myfile >> s;
            vacab.push_back(s);
        }
    }
    else cout << "vocab.txt can't open." << endl;
    myfile.close();
    myfile.open("unigram.txt");
    vector<int> count;
    int i;
    if (myfile.is_open()){
        while (!myfile.eof()){
            myfile >> i;
            count.push_back(i);
        }
    }
    else cout << "unigram.txt can't open." << endl;
    myfile.close();

    vector<pair<string, double> > unigram = uniDis(vacab, count, 'M');
    for (int j = 0; j < unigram.size(); j++){
        cout << unigram[j].first << ": " << unigram[j].second << endl;
    }

    vector<pair<int, int> > combIndex;
    vector<pair<int, int> > lookuptable;
    int n, m, p;
    myfile.open("bigram.txt");
    if (myfile.is_open()){
        string line;
        while (getline(myfile, line))
        {
            istringstream iss(line);
            if (!(iss >> n >> m >> p)) { cout << "no lin" << endl; break; }
            else{
                pair<int, int> pp(n, m);
                combIndex.push_back(pp);
                pair<int, int> ppp(n, p);
                lookuptable.push_back(ppp);
            }
        }
    }
    else cout << "bigram.txt can't open." << endl;
    myfile.close();

    vector<pair<string, double> > bigram = bigDis(vacab, count, "THE", 10,
        combIndex, lookuptable);
    for (int i = 0; i < bigram.size(); i++){
        cout << "1st most likely word: " << bigram[i].first << " with bigram
            probability: " << bigram[i].second << endl;
    }

    vector<string>
        sentence{"THE","STOCK","MARKET","FELL","BY","ONE","HUNDRED","POINTS","LAS
        T","WEEK"};
    vector<double> l = questionc(sentence, vacab, count, combIndex, lookuptable);
    cout << "log likelihood under unigram: " << l[0] << endl;
    cout << "log likelihood under bigram: " << l[1] << endl;
    vector<string>
        sentence1{"THE","SIXTEEN","OFFICIALS","SOLD","FIRE","INSURANCE"};
    vector<double> l1 = questionc(sentence1, vacab, count, combIndex, lookuptable
        );
    cout << "log likelihood under unigram: " << l1[0] << endl;
    cout << "log likelihood under bigram: " << l1[1] << endl;
    vector<double> lamda;
    for (double i = 0; i <= 1.01; i+=0.01){ lamda.push_back(i); }
    vector<double> lb2 = questiond(sentence1, vacab, count, combIndex,
        lookuptable, lamda);

}
```