2.1 Probabilistic reasoning

(a)

Simplify ratio r_k:

$$
\begin{aligned}
r_k &= \frac{P(D=0|S_1=1,S_2=1,\dots,S_k=1)}{P(D=1|S_1=1,S_2=1,\dots,S_k=1)} \\
&= \frac{P(D=0,S_1=1,S_2=1,\dots,S_k=1)}{P(S_1=1,S_2=1,\dots,S_k=1)} \Big/ \frac{P(D=1,S_1=1,S_2=1,\dots,S_k=1)}{P(S_1=1,S_2=1,\dots,S_k=1)} \\
&= \frac{P(D=0,S_1=1,S_2=1,\dots,S_k=1)}{P(D=1,S_1=1,S_2=1,\dots,S_k=1)} \\
&= \frac{P(D=0)\prod_{i=1}^{k}P(S_i=1|D=0)}{P(D=1)\prod_{i=1}^{k}P(S_i=1|D=1)} \\
&= \frac{\frac{1}{2}\prod_{i=1}^{k}(1-\frac{1}{2})}{\frac{1}{2}\times 1\times\prod_{i=2}^{k}\frac{f(i-1)}{f(i)}} \\
&= \frac{\frac{1}{2^{k+1}}}{\frac{1}{2}\frac{f(1)}{f(k)}} \\
&= \frac{2^k+(-1)^k}{2^k} \\
&= 1+(-\frac{1}{2})^k
\end{aligned}
$$

Therefore, when k is odd number, then $r_k<1$, which leads to $D=1$. When k is even number, then $r_k>1$, which leads to $D=0$. In conclusion:

$$
D=\begin{cases}0(k=2i-1)\\1(k=2i)\end{cases}
$$

(b)

In (a), we have calculated that:

$$
\begin{aligned}
P(D=1,S_1=1,S_2=1,\dots,S_k=1) &= \frac{1}{2}\frac{f(1)}{f(k)} \\
&= \frac{1}{2^{k+1}+2\times(-1)^k}
\end{aligned}
$$

Obviously when k is raising, then the $P(D=1,S_1=1,S_2=1,\dots,S_k=1)$ is decreasing. On the other hand, $P(D=0,S_1=1,S_2=1,\dots,S_k=1)$ is raising. Therefore, the diagnosis become more certain as more symptoms are observed.

2.2 Noisy-OR

(a) Conditional probability table:

The Noisy-OR of this network is:

$$P(Y=1|X_1=x_1,X_2=x_2,X_3=x_3)=1-(1-p_1)^{x_1}(1-p_2)^{x_2}(1-p_3)^{x_3}$$

Then according to the table:

$$\begin{cases} 1-(1-p_2)=\dfrac{1}{3} \\ 1-(1-p_2)(1-p3)=\dfrac{4}{5} \\ 1-(1-p_1)(1-p_2)(1-p3)=\dfrac{5}{6} \end{cases}$$

Then:

$$\begin{cases} p_1=\dfrac{1}{6} \\ p_2=\dfrac{1}{3} \\ p_3=\dfrac{7}{10} \end{cases}$$

Therefore, the completed table is as follows:

X_1	X_2	X_3	$P(Y=1\|X_1,X_2,X_3)$
0	0	0	0
1	0	0	$\frac{1}{6}$
0	1	0	$\frac{1}{3}$
0	0	1	$\frac{7}{10}$
1	1	0	$\frac{4}{9}$
1	0	1	$\frac{3}{4}$
0	1	1	$\frac{4}{5}$
1	1	1	$\frac{5}{6}$

(b) Qualitative reasoning

In question (b), we assume that the

$$P(X_2 = 1|Y = 0) = \frac{P(Y = 0|X_2 = 1)P(X_2 = 1)}{P(Y = 0)}$$

If $X_2 = 1$, it is less likely that $Y = 0$. Therefore, $P(Y = 0|X_2 = 1) < P(Y = 0)$. In conclusion:

$$P(X_2 = 1|Y = 0) < P(X_2 = 1)$$

Like above:

$$P(X_2 = 1|Y = 1) = \frac{P(Y = 1|X_2 = 1)P(X_2 = 1)}{P(Y = 1)}$$

If $X_2 = 1$, it is more likely that $Y = 1$. Therefore, $P(Y = 1|X_2 = 1) > P(Y = 1)$. In conclusion:

$$P(X_2 = 1|Y = 1) > P(X_2 = 1)$$

Consider $P(X_2 = 1|Y = 1, X_1 = 0, X_3 = 0)$, we can compare it with $P(X_2 = 1|Y = 1)$. If $X_1 = 0, X3 = 0$ but $Y = 1$, it is more likely that $X_2 = 1$. Therefore:

$$P(X_2 = 1|Y = 1, X_1 = 0, X_3 = 0) > P(X_2 = 1|Y = 1)$$

Consider $P(X_2 = 1|Y = 1, X_1 = 1, X_3 = 1)$, we can compare it with $P(X_2 = 1)$:

$$\begin{aligned} P(X_2 = 1|Y = 1, X_1 = 1, X_3 = 1) &= \frac{P(Y = 1|X_1 = 1, X_2 = 1, X_3 = 1)P(X_2 = 1|X_1 = 1, X_3 = 1)}{P(Y = 1|X_1 = 1, X_3 = 1)} \\ &= \frac{P(Y = 1|X_1 = 1, X_2 = 1, X_3 = 1)P(X_2 = 1)}{P(Y = 1|X_1 = 1, X_3 = 1)} \end{aligned}$$

Obviously $P(Y = 1|X_1 = 1, X_2 = 1, X_3 = 1) > P(Y = 1|X_1 = 1, X_3 = 1)$. In conclusion:

$$P(X_2 = 1|Y = 1, X_1 = 1, X_3 = 1) > P(X_2 = 1)$$

If $X_1 = 1, X_3 = 1$, it is more likely that $Y = 1$. Therefore:

$$P(X_2 = 1|Y = 1, X_1 = 1, X_3 = 1) < P(X_2 = 1|Y = 1)$$

In conclusion:

$$\begin{aligned} &P(X_2 = 1|Y = 0) < P(X_2 = 1) < P(X_2 = 1|Y = 1, X_1 = 1, X_3 = 1) \\ &< P(X_2 = 1|Y = 1) < P(X_2 = 1|Y = 1, X_1 = 0, X_3 = 0) \end{aligned}$$

(c)

The calculation of these probabilities are a little bit complex. Therefore, I write a program to help me to calculate them. The program is as follows.

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
#include <fstream>
#include <string>

using std::cin;
using std::cout;
using std::endl;
using std::vector;
using std::istream;
using std::ostream;
using std::ofstream;
using std::ifstream;
using std::string;

class NOR
{
public:
    NOR(): n(0) {}//默认构造函数
    NOR(istream& in)//流输入构造, 将给定该题数据的txt格式
    {
        //先输入n, 再依次为 graph, graphVar, base, baseVar
        in >> n;
        string tmpString;
        for (size_t i = 0; i < n; i++)
        {
            in >> tmpString;
            names.push_back(tmpString);
        }
        vector<int> tmpVecInt;
        vector<double> tmpVecDouble;
        int tmpInt;
        double tmpDouble;
        for (size_t i = 0; i < n; i++)
        {
            tmpVecInt.clear();
            for (size_t j = 0; j < n; j++)
            {
                in >> tmpInt;
                tmpVecInt.push_back(tmpInt);
            }
            graph.push_back(tmpVecInt);
        }
        for (size_t i = 0; i < n; i++)
        {
            tmpVecDouble.clear();
            for (size_t j = 0; j < n; j++)
            {
                in >> tmpDouble;
                tmpVecDouble.push_back(tmpDouble);
            }
            graphVar.push_back(tmpVecDouble);
        }
        for (size_t i = 0; i < n; i++)
        {
            in >> tmpInt;
            base.push_back(tmpInt);
        }
        for (size_t i = 0; i < n; i++)
        {
            in >> tmpDouble;
            baseVar.push_back(tmpDouble);
        }
    }
    ~NOR() {}//析构函数
    double PFull(vector<bool> p)//全概率计算
    {
        double pSum = 1, tmpP;
        int i, j;
        for (i = 0; i < n; i++)//根节点计算
        {
            if (base[i])
            {
                if (p[i]) pSum *= baseVar[i];
                else pSum *= (1 - baseVar[i]);
            }
        }
        for (i = 0; i < n; i++)//子节点计算
        {
            if (base[i]) continue;
            tmpP = 1;
            for (j = 0; j < n; j++)
            {
                if (graph[j][i] && p[j]) tmpP *= (1 - graphVar[j][i]);
            }
            if (p[i]) pSum *= (1 - tmpP);
            else pSum *= tmpP;
        }
        return pSum;
    }
    //a[i]=1代表该全概率生效, abP=1代表点亮, na用于存储概率表示形式
    double P(vector<bool> a, vector<bool> b, vector<bool> abP, string
&na)
    {
        double aSum = 0, bSum = 0;
        vector<vector<bool>> aPs = allP(a, abP), bPs = allP(b, abP);
        int i;
        for (i = 0; i < aPs.size(); i++) aSum += PFull(aPs[i]);
        for (i = 0; i < bPs.size(); i++) bSum += PFull(bPs[i]);
        na.clear();
        na += "P( ";
        for (i = 0; i < n; i++)
        {
            if (a[i])
            {
                na += names[i] + "=";
                if (abP[i]) na += "1 ";
                else na += "0 ";
            }
        }
        na += ") / P( ";
        for (i = 0; i < n; i++)
        {
            if (b[i])
            {
                na += names[i] + "=";
                if (abP[i]) na += "1 ";
                else na += "0 ";
            }
        }
        na += ")";
        return aSum / bSum;
    }
private:
    //用于返回一个bool值的所有组合
    vector<vector<bool>> allP(vector<bool> a, vector<bool> aP)
    {
        int i = 0, j = 0;
        vector<vector<bool>> ans;
        vector<bool> tmp;
        int sn = 0;
        for (i = 0; i < n; i++)
        {
            if (!a[i]) sn++;
        }
        long long s = 1, is, tmpis;
        for (i = 0; i < sn; i++)
        {
            s *= 2;
        }
        for (is = 0; is < s; is++)
        {
            tmp.clear();
            tmpis = is;
            for (j = 0; j < n; j++)
            {
                if (a[j]) tmp.push_back(aP[j]);
                else
                {
                    tmp.push_back(tmpis % 2);
                    tmpis /= 2;
                }
            }
            ans.push_back(tmp);
        }
        return ans;
    }
    int n;//一共n个节点
    vector<string> names;
    vector<vector<int>> graph;//graph[a][b]指代a指向b的图. 1代表a->b, 0代表
不连接
    vector<vector<double>> graphVar;//有向图的值, 若不连接默认为0
    vector<int> base;//根节点, 为根节点的值为1, 否则为0
    vector<double> baseVar;//根节点对应的值
};

int main()
{
    ifstream inData("data2_2.txt");
    NOR ans(inData);
    string name;
    double tmpDouble;
    //下列输出与问题a比较检查程序是否有误
    cout << "The code check(compare with question a):" << endl;
    tmpDouble = ans.P({1, 1, 1, 1}, {0, 1, 1, 1}, {1, 0, 0, 0}, name);
    cout << name << " = " << tmpDouble << endl;
    tmpDouble = ans.P({1, 1, 1, 1}, {0, 1, 1, 1}, {1, 1, 0, 0}, name);
    cout << name << " = " << tmpDouble << endl;
    tmpDouble = ans.P({1, 1, 1, 1}, {0, 1, 1, 1}, {1, 0, 1, 0}, name);
    cout << name << " = " << tmpDouble << endl;
    tmpDouble = ans.P({1, 1, 1, 1}, {0, 1, 1, 1}, {1, 0, 0, 1}, name);
    cout << name << " = " << tmpDouble << endl;
    tmpDouble = ans.P({1, 1, 1, 1}, {0, 1, 1, 1}, {1, 1, 1, 0}, name);
    cout << name << " = " << tmpDouble << endl;
    tmpDouble = ans.P({1, 1, 1, 1}, {0, 1, 1, 1}, {1, 1, 0, 1}, name);
    cout << name << " = " << tmpDouble << endl;
    tmpDouble = ans.P({1, 1, 1, 1}, {0, 1, 1, 1}, {1, 0, 1, 1}, name);
    cout << name << " = " << tmpDouble << endl;
    tmpDouble = ans.P({1, 1, 1, 1}, {0, 1, 1, 1}, {1, 1, 1, 1}, name);
    cout << name << " = " << tmpDouble << endl;
    //输出条件概率
    cout << endl << "The answer of question b:" << endl;
    tmpDouble = ans.P({0, 0, 1, 0}, {0, 0, 0, 0}, {0, 0, 1, 0}, name);
    cout << name << " = " << tmpDouble << endl;
    tmpDouble = ans.P({1, 0, 1, 0}, {1, 0, 0, 0}, {0, 0, 1, 0}, name);
    cout << name << " = " << tmpDouble << endl;
    tmpDouble = ans.P({1, 0, 1, 0}, {1, 0, 0, 0}, {1, 0, 1, 0}, name);
    cout << name << " = " << tmpDouble << endl;
    tmpDouble = ans.P({1, 1, 1, 1}, {1, 1, 0, 1}, {1, 0, 1, 0}, name);
    cout << name << " = " << tmpDouble << endl;
    tmpDouble = ans.P({1, 1, 1, 1}, {1, 1, 0, 1}, {1, 1, 1, 1}, name);
    cout << name << " = " << tmpDouble << endl;
    return 0;
}
```

And the "data2_2.txt" is as follows:

```
4

Y   X1  X2  X3

0   0   0   0
1   0   0   0
1   0   0   0
1   0   0   0

0   0   0   0
0.166667    0   0   0
0.333333    0   0   0
0.7 0   0   0

0   1   1   1

0   0.333333    0.333333    0.333333
```

Because it is not easy to express fraction in C++, I only use float to express fraction. For example, $\frac{1}{3} \approx 0.333333$.

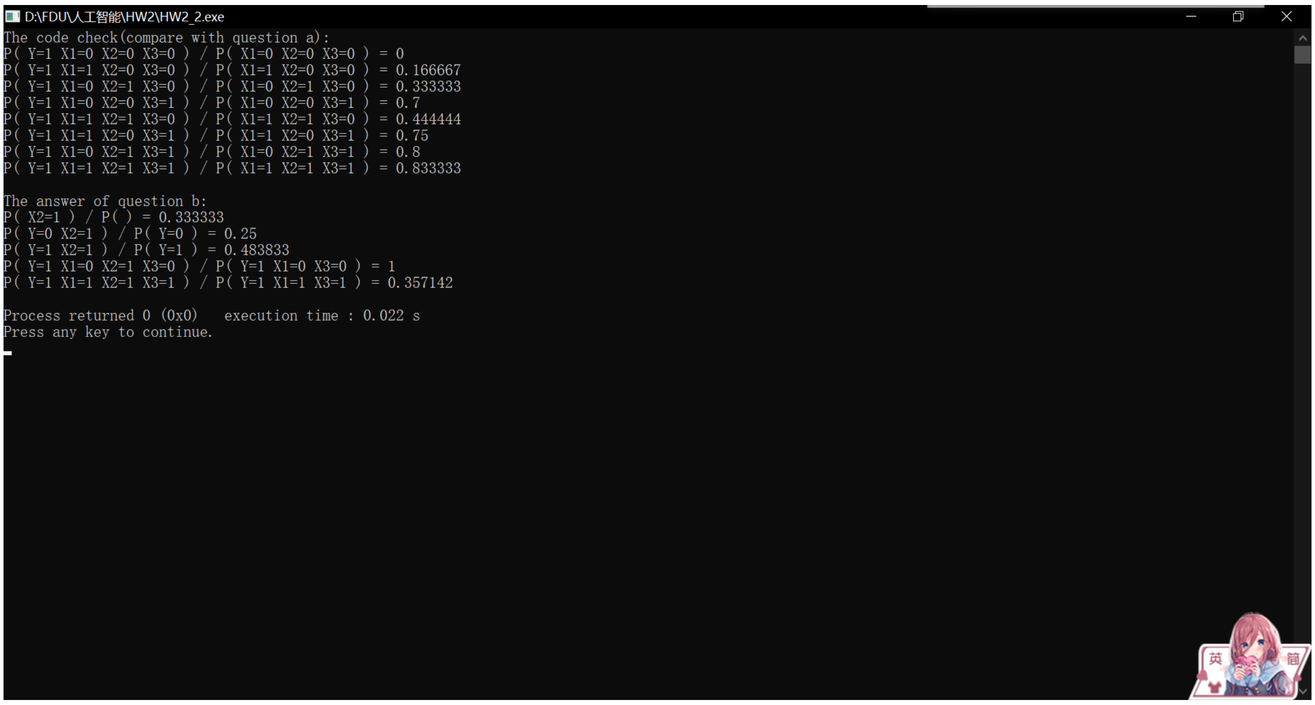
The answer is:

```
The code check(compare with question a):
P( Y=1 X1=0 X2=0 X3=0 ) / P( X1=0 X2=0 X3=0 ) = 0
P( Y=1 X1=1 X2=0 X3=0 ) / P( X1=1 X2=0 X3=0 ) = 0.166667
P( Y=1 X1=0 X2=1 X3=0 ) / P( X1=0 X2=1 X3=0 ) = 0.333333
P( Y=1 X1=0 X2=0 X3=1 ) / P( X1=0 X2=0 X3=1 ) = 0.7
P( Y=1 X1=1 X2=1 X3=0 ) / P( X1=1 X2=1 X3=0 ) = 0.444444
P( Y=1 X1=1 X2=0 X3=1 ) / P( X1=1 X2=0 X3=1 ) = 0.75
P( Y=1 X1=0 X2=1 X3=1 ) / P( X1=0 X2=1 X3=1 ) = 0.8
P( Y=1 X1=1 X2=1 X3=1 ) / P( X1=1 X2=1 X3=1 ) = 0.833333

The answer of question b:
P( X2=1 ) / P( ) = 0.333333
P( Y=0 X2=1 ) / P( Y=0 ) = 0.25
P( Y=1 X2=1 ) / P( Y=1 ) = 0.483833
```

```
P( Y=1 X1=0 X2=1 X3=0 ) / P( Y=1 X1=0 X3=0 ) = 1
P( Y=1 X1=1 X2=1 X3=1 ) / P( Y=1 X1=1 X3=1 ) = 0.357142
```

Shootscreen:



```
The code check(compare with question a):
P( Y=1 X1=0 X2=0 X3=0 ) / P( X1=0 X2=0 X3=0 ) = 0
P( Y=1 X1=1 X2=0 X3=0 ) / P( X1=1 X2=0 X3=0 ) = 0.166667
P( Y=1 X1=0 X2=1 X3=0 ) / P( X1=0 X2=1 X3=0 ) = 0.333333
P( Y=1 X1=0 X2=0 X3=1 ) / P( X1=0 X2=0 X3=1 ) = 0.7
P( Y=1 X1=1 X2=1 X3=0 ) / P( X1=1 X2=1 X3=0 ) = 0.444444
P( Y=1 X1=1 X2=0 X3=1 ) / P( X1=1 X2=0 X3=1 ) = 0.75
P( Y=1 X1=0 X2=1 X3=1 ) / P( X1=0 X2=1 X3=1 ) = 0.8
P( Y=1 X1=1 X2=1 X3=1 ) / P( X1=1 X2=1 X3=1 ) = 0.833333

The answer of question b:
P( X2=1 ) / P( ) = 0.333333
P( Y=0 X2=1 ) / P( Y=0 ) = 0.25
P( Y=1 X2=1 ) / P( Y=1 ) = 0.483833
P( Y=1 X1=0 X2=1 X3=0 ) / P( Y=1 X1=0 X3=0 ) = 1
P( Y=1 X1=1 X2=1 X3=1 ) / P( Y=1 X1=1 X3=1 ) = 0.357142

Process returned 0 (0x0)   execution time : 0.022 s
Press any key to continue.
```

If we check the first part, we can find that the program is right.

Therefore, keep four decimals:

$$
\begin{aligned}
&P(X_2=1|Y=0)=0.2500\\
&P(X_2=1)=0.3333\\
&P(X_2=1|Y=1,X_1=1,X_3=1)=0.3571\\
&P(X_2=1|Y=1)=0.4838\\
&P(X_2=1|Y=1,X_1=0,X_3=0)=1.0000
\end{aligned}
$$

In conclusion:

$$
\begin{aligned}
&P(X_2=1|Y=0)<P(X_2=1)<P(X_2=1|Y=1,X_1=1,X_3=1)\\
&<P(X_2=1|Y=1)<P(X_2=1|Y=1,X_1=0,X_3=0)
\end{aligned}
$$

It is as same as we have calculated in question (b).

In the mean time, the class NOR in my code can be used as a standard module.

2.3 Hangman

The code is as follows, some relevant notes are contained in it:

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <fstream>
#include <string>
```

```
using std::cin;
using std::cout;
using std::endl;
using std::vector;
using std::istream;
using std::ostream;
using std::ofstream;
using std::ifstream;
using std::string;

//定义的HangmanModule类仅能用于此题计算，而非通用bayesnet模板
class HangmanModule
{
public:
    HangmanModule(): sum(0) {}//默认构造函数
    //流输入构造，用于读取文件hw2_word_counts_05.txt并初始化参数列表
    HangmanModule(istream& in): sum(0)
    {
        string w;
        long long tmp;
        int i, j;
        while (in >> w)
        {
            words.push_back(w);
            in >> tmp;
            times.push_back(tmp);
            sum += tmp;
        }
        for (i = 0; i < words.size(); i++)
        {
            for (j = i + 1; j < words.size(); j++)
            {
                if (times[j] < times[i])
                {
                    tmp = times[j];
                    times[j] = times[i];
                    times[i] = tmp;
                    w = words[j];
                    words[j] = words[i];
                    words[i] = w;
                }
            }
        }
        for (i = 0; i < times.size(); i++)
        {
            p.push_back(((long double) times[i]) / sum);
        }
    }
    /*
    计算题b的成员函数, 其中ans用于保存最大概率的那个字母,a为correctly guessed
    中的参数列表, 其中'-'用'\0'表示以便进行bool运算; notIn为incorrectly
guessed
    中的参数列表
    */
    long double pE(char& ans, vector<char> a, vector<char> notIn)
    {
        /*
        subW用于保存样式为correctly guessed中的单词, 而subT以及subP为其对应的
        次数和频率, alphaBetP用于存储猜测符合条件的单词的次数
        */
        vector<string> subW;
        vector<long long> subT, alphaBetP;
        vector<long double> subP;
        //alphaBet为顺序26个字母表, 声明为静态变量以减小增删开销
        static string alphaBet("ABCDEFGHIJKLMNOPQRSTUVWXYZ");
        long long tmpLI = 0, tmpFI = 0;
        int i, j, k;
        bool pin;
        /*
        将correctly guessed中猜对的字母纳入notIn, 代表在后面的猜测中不需要考虑
        已经猜对的正确字母
        */
        for (i = 0; i < 5; i++)
        {
            if (a[i]) notIn.push_back(a[i]);
        }
        //筛选符合条件的单词, 初始化subW, subT, subP
        for (i = 0; i < words.size(); i++)
        {
            pin = true;
            for (j = 0; j < 5; j++)
            {
                if (a[j])
                {
                    if (a[j] != words[i][j])
                    {
                        pin = false;
                        break;
                    }
                }
                else
                {
                    for (k = 0; k < notIn.size(); k++)
                    {
                        if (words[i][j] == notIn[k])
                        {
                            pin = false;
                            break;
                        }
                    }
                }
            }
            if (pin)
            {
                subW.push_back(words[i]);
                subT.push_back(times[i]);
                subP.push_back(p[i]);
            }
        }
        //计算alphaBetP
        for (i = 0; i < subT.size(); i++)
        {
            tmpFI += subT[i];
        }
        for (i = 0; i < 26; i++)
        {
            pin = true;
            for (j = 0; j < notIn.size(); j++)
            {
                if (alphaBet[i] == notIn[j])
                {
                    pin = false;
                    break;
                }
            }
            if (!pin)
            {
                alphaBetP.push_back(0);
                continue;
            }
            tmpLI = 0;
            for (j = 0; j < subW.size(); j++)
            {
                pin = false;
                for (k = 0; k < 5; k++)
                {
                    if (!a[k])
                    {
                        if (alphaBet[i] == subW[j][k])
                        {
                            pin = true;
                            break;
                        }
                    }
                }
                if (pin) tmpLI += subT[j];
            }
            alphaBetP.push_back(tmpLI);
        }
        tmpLI = alphaBetP[0];
        j = 0;
        //找出符合条件的最大alphaBetP成员
        for (i = 0; i < alphaBetP.size(); i++)
        {
            if (alphaBetP[i] > tmpLI)
            {
                j = i;
                tmpLI = alphaBetP[i];
            }
        }
        ans = alphaBet[j];//将引用值归为最大的那个字母
        return ((long double)tmpLI) / tmpFI;
    }
    //下列三个为访问器函数
    vector<string> Words()
    {
        return words;
    }
    vector<long long> Times()
    {
        return times;
    }
    vector<long double> P()
    {
        return p;
    }
    //析构函数
    ~HangmanModule() {}
private:
    /*
    words存储使用频率由少到多的所有单词
    times存储对应words中单词的出现次数，使用长整型防止溢出
    p存储对应words中单词的频率，其中使用长双精度防止截断误差
    sum存储总单词数，使用长整型防止溢出
    */
    vector<string> words;
    vector<long long> times;
    vector<long double> p;
    long long sum;
};

int main()
{
    long long sum = 0, tmp;
    ifstream data("hw2_word_counts_05.txt");
    HangmanModule ans(data);
    vector<string> w;
    vector<long long> t;
    vector<long double> p;
    w = ans.Words();
    t = ans.Times();
    p = ans.P();
    cout.precision(4);//将输出浮点数精度设置为4
    //输出频率最高的十个单词及其相应的次数和频率
    cout << "The Most" << endl << "Word\tCount\tProbability" << endl;
    for (int i = w.size() - 1; i > w.size() - 11; i--)
    {
        cout << w[i] << "\t" << t[i] << "\t" << p[i] << endl;
    }
    cout << endl;
    //输出频率最低的十个单词及其对应的次数和频率
    cout << "The Least" << endl << "Word\tCount\tProbability" << endl;
    for (int i = 0; i < 10; i++)
    {
        cout << w[i] << "\t" << t[i] << "\t" << p[i] << endl;
    }
    cout << endl;
    //ansDouble暂存概率, ansChar暂存概率最大的字母(未编写排版函数, 所以有些混乱)
    long double ansDouble;
    char ansChar;
    cout << "Guess\tIncorrect\tBest\tP" << endl;
    ansDouble = ans.pE(ansChar, {'\0', '\0', '\0', '\0', '\0'}, {});
    cout << "-----\t{}\t\t" << ansChar << '\t' << ansDouble << endl;
    ansDouble = ans.pE(ansChar, {'\0', '\0', '\0', '\0', '\0'}, {'E',
'O'});
    cout << "-----\t{E,O}\t\t" << ansChar << '\t' << ansDouble << endl;
    ansDouble = ans.pE(ansChar, {'Q', '\0', '\0', '\0', '\0'}, {});
    cout << "Q----\t{}\t\t" << ansChar << '\t' << ansDouble << endl;
    ansDouble = ans.pE(ansChar, {'Q', '\0', '\0', '\0', '\0'}, {'U'});
    cout << "Q----\t{U}\t\t" << ansChar << '\t' << ansDouble << endl;
    ansDouble = ans.pE(ansChar, {'\0', '\0', 'Z', 'E', '\0'}, {'A', 'D',
'I', 'R'});
    cout << "--ZE-\t{A,D,I,R}\t" << ansChar << '\t' << ansDouble <<
endl;
    ansDouble = ans.pE(ansChar, {'\0', '\0', '\0', '\0', '\0'}, {'E',
'O'});
    cout << "-----\t{E,O}\t\t" << ansChar << '\t' << ansDouble << endl;
    ansDouble = ans.pE(ansChar, {'D', '\0', '\0', 'I', '\0'}, {});
    cout << "D--I-\t{}\t\t" << ansChar << '\t' << ansDouble << endl;
    ansDouble = ans.pE(ansChar, {'D', '\0', '\0', 'I', '\0'}, {'A'});
    cout << "D--I-\t{A}\t\t" << ansChar << '\t' << ansDouble << endl;
    ansDouble = ans.pE(ansChar, {'\0', 'U', '\0', '\0', '\0'}, {'A',
'E', 'I', 'O', 'S'});
```

```
    cout << "-U---\t{A,E,I,O,S}\t" << ansChar << '\t' << ansDouble <<
endl;
    return 0;
}
```

And the answer is:

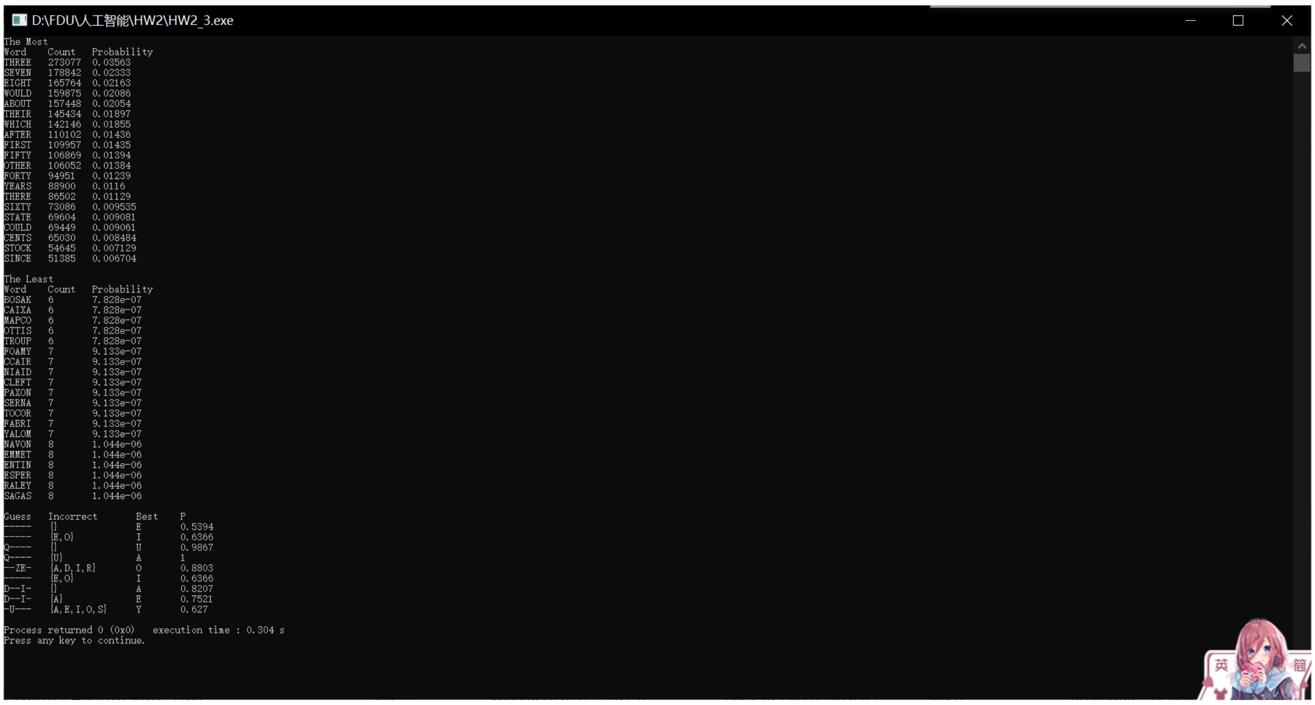
```
The Most
Word    Count   Probability
THREE   273077  0.03563
SEVEN   178842  0.02333
EIGHT   165764  0.02163
WOULD   159875  0.02086
ABOUT   157448  0.02054
THEIR   145434  0.01897
WHICH   142146  0.01855
AFTER   110102  0.01436
FIRST   109957  0.01435
FIFTY   106869  0.01394
OTHER   106052  0.01384
FORTY   94951   0.01239
YEARS   88900   0.0116
THERE   86502   0.01129
SIXTY   73086   0.009535
STATE   69604   0.009081
COULD   69449   0.009061
CENTS   65030   0.008484
STOCK   54645   0.007129
SINCE   51385   0.006704

The Least
Word    Count   Probability
BOSAK   6       7.828e-07
CAIXA   6       7.828e-07
MAPCO   6       7.828e-07
OTTIS   6       7.828e-07
TROUP   6       7.828e-07
FOAMY   7       9.133e-07
CCAIR   7       9.133e-07
NIAID   7       9.133e-07
CLEFT   7       9.133e-07
PAXON   7       9.133e-07
SERNA   7       9.133e-07
TOCOR   7       9.133e-07
FABRI   7       9.133e-07
YALOM   7       9.133e-07
NAVON   8       1.044e-06
EMMET   8       1.044e-06
ENTIN   8       1.044e-06
```

```
ESPER    8       1.044e-06
RALEY    8       1.044e-06
SAGAS    8       1.044e-06

Guess    Incorrect       Best    P
-----    {}              E       0.5394
-----    {E,O}           I       0.6366
Q----    {}              U       0.9867
Q----    {U}             A       1
--ZE-    {A,D,I,R}       O       0.8803
-----    {E,O}           I       0.6366
D--I-    {}              A       0.8207
D--I-    {A}             E       0.7521
-U---    {A,E,I,O,S}     Y       0.627
```

Shootscreen:



```
The Most
Word    Count   Probability
THREE   273077  0.03563
SEVEN   178842  0.02333
EIGHT   165764  0.02163
WOULD   159875  0.02086
ABOUT   157448  0.02054
THEIR   145434  0.01897
WHICH   142146  0.01855
AFTER   110102  0.01436
FIRST   109957  0.01435
FIFTY   106869  0.01394
OTHER   106052  0.01384
FORTY   94951   0.01239
YEARS   88900   0.0116
THERE   86502   0.01129
SIXTY   73086   0.009535
STATE   69604   0.009081
COULD   69449   0.009061
CENTS   65030   0.008484
STOCK   54645   0.007129
SINCE   51385   0.006704

The Least
Word    Count   Probability
BOSAK   6       7.828e-07
CAIXA   6       7.828e-07
MAPCO   6       7.828e-07
OTTIS   6       7.828e-07
TROUP   6       7.828e-07
FOAMY   7       9.133e-07
CCAIR   7       9.133e-07
NIAID   7       9.133e-07
CLEFT   7       9.133e-07
PAXON   7       9.133e-07
SERNA   7       9.133e-07
TOCOR   7       9.133e-07
FABRI   7       9.133e-07
YALOM   7       9.133e-07
NAVON   8       1.044e-06
EMMET   8       1.044e-06
ENTIN   8       1.044e-06
ESPER   8       1.044e-06
RALEY   8       1.044e-06
SAGAS   8       1.044e-06

Guess   Incorrect       Best    P
-----   {}              E       0.5394
-----   {E,O}           I       0.6366
Q----   {}              U       0.9867
Q----   {U}             A       1
--ZE-   {A,D,I,R}       O       0.8803
-----   {E,O}           I       0.6366
D--I-   {}              A       0.8207
D--I-   {A}             E       0.7521
-U---   {A,E,I,O,S}     Y       0.627

Process returned 0 (0x0)   execution time : 0.304 s
Press any key to continue.
```

Therefore, the answer is:

(a)

i. The most frequent: THREE, SEVEN, EIGHT, WOULD, ABOUT, THEIR, WHICH, AFTER, FIRST, FIFTY.

ii. The least frequent(actually more than 10 words): BOSAK, CAIXA, MAPCO, OTTIS, TROUP, FOAMY, CCAIR, NIAID, CLEFT, PAXON, SERNA, TOCOR, FABRI, YALOM.

iii. I think it make sense. Because the most frequent words we are all familiar with, and the least frequent words are not.

(b)

correctly guessed	incorrectly guessed	best next guess l	$P(L_i = l\|E)$
-----	{}	E	0.5394
-----	{E,O}	I	0.6366
Q----	{}	U	0.9867
Q----	{U}	A	1.0000
---ZE-	{A,D,I,R}	O	0.8803
-----	{E,O}	I	0.6366
D--I-	{}	A	0.8207
D--I-	{A}	E	0.7521
-U---	{A,E,I,O,S}	Y	0.6270

2.4 Conditional independence

The answer is:

false, true, false, false, false, false, true, false, true, true.

(a) false;

Assume that:

$$X = \{F\}, Y = \{C\}, E = \{H\}$$

The paths between X and Y are:

$$C \Rightarrow F$$
$$F \Rightarrow H \Leftarrow C$$

The first path is not belongs to 3 conditions. Therefore, X and Y are not blocked by E . In conclusion:

$$P(F|H) \neq P(F|C, H)$$

(b) true;

Assume that:

$$X = \{E\}, Y = \{F\}, E = \{A, B\}$$

The paths between X and Y are:

$$E \Leftarrow B \Rightarrow F$$
$$E \Rightarrow G \Leftarrow F$$

The first path obey the condition 2 and the second path obey the condition 3. Therefore, X and Y are blocked by E. In conclusion:

$$P(E|A,B) = P(E|A,B,F)$$

(c) false;

Assume that:

$$X = \{E\}, Y = \{F\}, E = \{B, G\}$$

The paths between X and Y are:

$$E \Leftarrow B \Rightarrow F$$
$$E \Rightarrow G \Leftarrow F$$

The second path does not obey 3 conditions. Therefore, X and Y are not blocked by E . In conclusion:

$$P(E,F|B,G) \neq P(E|B,G)P(F|B,G)$$

(d) false;

Assume that:

$$X = \{F\}, Y = \{E\}, E = \{B, C, G, H\}$$

The paths between X and Y are:

$$E \Leftarrow B \Rightarrow F$$
$$E \Rightarrow G \Leftarrow F$$

The second path does not obey 3 conditions. Therefore, X and Y are not blocked by E . In conclusion:

$$P(F|B,C,G,H) \neq P(F|B,C,E,G,H)$$

(e) false;

Assume that:

$$X = \{A, B\}, Y = \{G, H\}, E = \{D, E, F\}$$

The paths between X and Y are too many. Therefore, I only list the one of not qualified paths:

$$A \Rightarrow E \Leftarrow B \Rightarrow F \Leftarrow C \Rightarrow H$$

The second path does not obey 3 conditions. Therefore, X and Y are not blocked by E . In conclusion:

$$P(F|B,C,G,H) \neq P(F|B,C,E,G,H)$$

(f) false;

$$P(D,E,F)=P(D)P(E|D)P(F|E,D)$$

Therefore, we have to judge if

$$P(F|E,D)=P(F|E)$$

Assume that:

$$X=\{F\},Y=\{D\},E=\{E\}$$

The paths between X and Y are:

$$\begin{gathered} F\Leftarrow B\Rightarrow E\Leftarrow A\Rightarrow D \\ F\Rightarrow G\Leftarrow E\Leftarrow A\Rightarrow D \end{gathered}$$

The first path dose not obey 3 conditions. Therefore, X and Y are not blocked by E. In conclusion:

$$P(D,E,F)\neq P(D)P(E|D)P(F|E)$$

(g) true;

Assume that:

$$X=\{A\},Y=\{F\},E=\{\}$$

The paths between X and Y are:

$$\begin{gathered} A\Rightarrow E\Leftarrow B\Rightarrow F \\ A\Rightarrow E\Rightarrow G\Leftarrow F \end{gathered}$$

They are all obey the condition 3. Therefore, X and Y are blocked by E. In conclusion:

$$P(A|F)=P(A)$$

(h) false;

Assume that:

$$X=\{E\},Y=\{F\},E=\{\}$$

The paths between X and Y are:

$$\begin{gathered} E\Leftarrow B\Rightarrow F \\ E\Rightarrow G\Leftarrow F \end{gathered}$$

The first path does not obey 3 conditions. Therefore, X and Y are not blocked by E. In conclusion:

$$P(E,F)\neq P(E)P(F)$$

(i) true;

Assume that:

$$X = \{D\}, Y = \{E\}, E = \{A\}$$

The path between X and Y is:

$$D \Leftarrow A \Rightarrow E$$

The path obey condition 2. Therefore, X and Y are blocked by E. In conclusion:

$$P(D|A) = P(D|A, E)$$

(j) true;

Assume that:

$$X = \{B\}, Y = \{C\}, E = \{\}$$

The paths between X and Y are:

$$\begin{gathered} B \Rightarrow F \Leftarrow C \\ B \Rightarrow F \Rightarrow H \Leftarrow C \\ B \Rightarrow E \Rightarrow G \Leftarrow F \Leftarrow C \\ B \Rightarrow E \Rightarrow G \Leftarrow F \Rightarrow H \Leftarrow C \end{gathered}$$

They all obey the condition 3. Therefore, X and Y are blocked by E. In conclusion:

$$P(B, C) = P(B)P(C)$$

2.5 Subsets

(a) $\{\}$

Because the path between A and other nodes does not contain the structure:

$$A = \ldots \Rightarrow Z \Leftarrow \ldots = Node$$

Therefore, there is no node fit the formula.

(b) $\{C, F\}$

The paths between A and F are:

$$\begin{gathered} A \Rightarrow C \Rightarrow F \\ A \Rightarrow B \Rightarrow E \Leftarrow C \Rightarrow F \end{gathered}$$

Therefore, A and F are blocked by C. In conclusion:

$$P(A|C) = P(A|C, F)$$

(c) $\{B, C, D, E, F\}$

If we assume that:

$$X = \{A\}, Y = \{D, E, F\}, E = \{B, C\}$$

And list all the paths between X and Y. We can find that they all have structure as follows:

$$X = \ldots \Rightarrow E \Rightarrow \ldots = Y$$

They all obey condition 2. Therefore:

$$P(A|B,C) = P(A|B,C,D,E,F)$$

(d) $\{\}$

Because the path between B and other nodes does not contain the structure:

$$B = \ldots \Rightarrow Z \Leftarrow \ldots = Node$$

Therefore, there is no node fit the formula.

(e) $\{A, E\}$

If we add A, E in S, we can not find any nodes fit 3 conditions. Therefore, the largest subset is:

$$S = \{A, E\}$$

(f) $\{A, C, E, F\}$

If we add A, C, E in S, the node F fit condition 1 and 2. Other nodes can not fit 3 conditions. Therefore:

$$S = \{A, C, E, F\}$$

In conclusion:

$$P(B|A,C,E) = P(B|A,C,E,F)$$

(g) $\{E, F\}$

Because the path between D and E, F contains the structure:

$$D = \ldots \Rightarrow Z \Leftarrow \ldots = Node$$

And other nodes does not fit the path upon. Therefore:

$$S = \{E, F\}$$

In conclusion:

$$P(D) = P(D|E,F)$$

(h) $\{A, C, F\}$

If we add A in S, then C fit the condition 2 and 3, F fit the condition 2 and 3. Other nodes does not fit the 3 conditions. Therefore:

$$S = \{A, C, F\}$$

In conclusion:

$$P(D|A) = P(D|A, C, F)$$

(i) $\{C, E, F\}$

If we add C, E in S, then F fit the condition 1 and 2. Other nodes does not fit the 3 conditions. Therefore:

$$S = \{C, E, F\}$$

In conclusion:

$$P(D|C, E) = P(D|C, E, F)$$

(j) $\{E, F\}$

Actually we only need find the node fit the structure as follows:

$$D = \ldots \Rightarrow Z \Leftarrow \ldots = Node$$

Therefore, E, F are qualified. In conclusion:

$$S = \{E, F\}$$

Therefore:

$$P(D|F) = P(D|E, F)$$