如果背了N个词根，那这些词根到底会不会在单词里出现呢。更确切的描述是：长度不超过L，只由小写字母组成的，至少包含一个词根的单词，一共可能有多少个呢？

本题目包含多组数据，请处理到文件结束。每组数据占两行。  
第一行有两个正整数N和L。(0<N<6,0<L<2^31)  
第二行有N个词根，每个词根仅由小写字母组成，长度不超过5。两个词根中间用一个空格分隔开。对于每组数据，请在一行里输出一共可能的单词数目。  
由于结果可能非常巨大，你只需要输出单词总数模2^64的值。

//unsingned long long自动对2^64取模

typedef long long ll;

typedef unsigned long long ull;

const int maxn = 35;

int m, n;

int x;

struct Trie

{

ull mat[maxn][maxn];

int nxt[maxn][26], fail[maxn], end[maxn];

//nxt为字典树, fail相当于kmp中的nxt数组, end对单词结尾做标记

int root, L; //L相当于字典树中的sz ，root为根节点（即；0）

int newnode() //初相当于始化一个字典树上的节点

{

for (int i = 0;i < 26;i++) nxt[L][i] = -1;

end[L++] = 0;

return L - 1;

}

void init()

{

L = 0;

root = newnode();

memset(end, 0, sizeof(end));

}

void insert(char buf[]) //大致于字典树插入过程相同

{

int len = strlen(buf);

int now = root;

for (int i = 0;i < len;i++)

{

int x = buf[i] - 'a';

if (nxt[now][x] == -1) nxt[now][x] = newnode();

now = nxt[now][x];

}

end[now]++; //在单词结尾处做标记

}

void build() //相当于kmp的操作

{

queue<int>que;

fail[root] = root; //根节点初始化为0(即其本身)

for (int i = 0;i < 26;i++)

{

if (nxt[root][i] == -1) nxt[root][i] = root;

else //Trie中已经构建的节点

{

int x = nxt[root][i];

fail[x] = root;

que.push(x);

}

}

while (!que.empty())

{

int now = que.front();

que.pop();

if (end[fail[now]])end[now]++;

for (int i = 0;i < 26;i++)

{

if (nxt[now][i] == -1) //无后继点

nxt[now][i] = nxt[fail[now]][i];//类似于kmp中求nxt数组一样

else //存在下一个节点

{

int x = nxt[now][i];

fail[x] = nxt[fail[now]][i];

//失配指针指向他父节点的失配指针的下一个相同字符处

que.push(x);

}

}

}

}

void buildjz()

{

for (int i = 0;i < L;i++)

for (int j = 0;j < 26;j++)

if (!end[i] && !end[nxt[i][j]])

mat[i][nxt[i][j]]++;

for (int i = 0;i <= L;i++)

mat[i][L] = 1;//构造求平方和的矩阵

}

int query(char buf[]) //相当于字典树中的访问操作

{

int len = strlen(buf);

int now = root;

int res = 0;

for (int i = 0;i < len;i++)

//沿着整个文本串移动,每移动到一个字符（节点） 时，通过失配指针不断找寻模式串 ，重点为源头,找到一个就将其标记清除

{

now = nxt[now][buf[i] - 'a'];

int tmp = now;

while (tmp != root)

{

res += end[tmp];

end[tmp] = 0;

tmp = fail[tmp];

}

}

return res; //返回单词个数

}

ull res[maxn][maxn], tmp[maxn][maxn];

void mul(ull a[][maxn], ull b[][maxn], ll l)

{

for (int i = 0;i < l;i++)

for (int j = 0;j < l;j++)

{

tmp[i][j] = 0;

for (int k = 0;k < l;k++)

tmp[i][j] = (tmp[i][j] + a[i][k] \* b[k][j]);

}

for (int i = 0;i < l;i++)

for (int j = 0;j < l;j++)

a[i][j] = tmp[i][j];

}

void pow(ll k, ll l)

{

memset(res, 0, sizeof(res));

for (int i = 0;i < l;i++)

res[i][i] = 1;

while (k)

{

if (k & 1) mul(res, mat, l);

mul(mat, mat, l);

k /= 2;

}

}

void debug()

{

for (int i = 0;i < L;i++)

{

printf("%id=%3d,fail=%3d,end=%3d,chi=[", i, fail[i], end[i]);

for (int j = 0;j < 26;j++) printf("%2d", nxt[i][j]);

printf("]\n");

}

}

};

Trie ac;

char buf[12];

int main()

{

ull ans, b;

while (scanf("%d%d", &m, &n) != EOF)

{

memset(ac.mat, 0, sizeof(ac.mat));

ac.init();

for (int i = 0;i < m;i++)

{

scanf("%s", buf);

ac.insert(buf);

}

ans = 0;

ac.build();

ac.buildjz();

ac.pow(n, ac.L + 1);

for (int i = 0;i <= ac.L;i++)

ans = ans + ac.res[0][i];

ac.mat[0][0] = 26;

ac.mat[0][1] = 1;

ac.mat[1][1] = 1;

ac.mat[1][0] = 0;

ac.pow(n + 1, 2);//求26^0-26^n之和

b = ac.res[0][1];

b = b - ans;

printf("%I64u\n", b);

}

return 0;

}