单词接龙

题目描述：

单词接龙的规则是：可用于接龙的单词首字母必须要前一个单词的尾字母相同；当存在多个首字母相同的单词时，取长度最长的单词，如果长度也相等，则取字典序最小的单词；已经参与接龙的单词不能重复使用。
现给定一组全部由小写字母组成单词数组，并指定其中的一个单词作为起始单词，进行单词接龙，请输出最长的单词串，单词串是单词拼接而成，中间没有空格。

输入描述：

输入的第一行为一个非负整数，表示起始单词在数组中的索引 K，0 <= K < N ；

输入的第二行为一个非负整数，表示单词的个数 N；

接下来的 N 行，分别表示单词数组中的单词。

输出描述：

输出一个字符串，表示最终拼接的单词串。

补充说明：

单词个数 N 的取值范围为[1, 20]；

单个单词的长度的取值范围为[1, 30]；

示例 1

输入：

```
0

6

word

dd

da

dc

dword

d
```

输出：

```
worddwordda
```

说明：

先确定起始单词 word，再接以 d 开头的且长度最长的单词 dword，剩余以 d 开头且长度最长的有 dd、da、dc，则取字典序最小的 da，所以最后输出 worddwordda。

示例 2

输入：

```
4
6
word
dd
da
dc
dword
d
```

输出：

```
dwordda
```

说明：

先确定起始单词 dword，剩余以 d 开头且长度最长的有 dd、da、dc，则取字典序最小的 da，所以最后输出 dwordda。

```
#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

using namespace std;


bool comparestring(string a, string b) {
```

```
    if(a.length() != b.length()) return a.length() > b.length();
    else return a < b;
}

string maxwords(vector<string>& words, int k, int n) {
    if(n == 0) return "";
    string begin_word = words[k];
    char ch = begin_word.back();
    words.erase(words.begin()+k);

    string ans = begin_word;
    while(!words.empty()) {
        vector<string> lists;
        for(auto word : words) {
            if(word[0] == ch) {
                lists.push_back(word);
            }
        }
        if(!lists.empty()) {
            sort(lists.begin(), lists.end(), comparestring);
            ans += lists[0];
            words.erase(find(words.begin(), words.end(), lists[0]));
```

```
            ch = lists[0].back();
        } else break;
    }
    return ans;

}

int main() {
    int k, n;
    cin >> k;
    cin >> n;
    vector<string> words;
    string str;
    for(int i = 0; i < n; ++i) {
        cin >> str;
        words.push_back(str);
    }
    string res = maxwords(words, k, n);
    cout << res << endl;
    return 0;
}
// 64 位输出请用 printf("%lld")
```