Before we begin, it helps to *preprocess* the input, so that the schools are numbered instead of identified by strings.

Additionally, since what matters is what school each student comes from, it helps to form a table `schools` where `schools[i]` is a table containing the indices of the students hailing from school i. In Python, this can be accomplished through a nested list, and in C++ using nested vectors. In the implementations below, note that we account for how in the final output, the students are indexed starting from 1.

Creating this table can be done in $O(n)$ time.

In C++, you can use a `map` to associate the names of schools with integers.

```cpp
map<string,int> name_to_int;
vector<vector<int>> schools;
int n, m; cin>>n>>m;
for (int i = 0; i < n; i++)
{
    string s; cin>>s;
    if (name_to_int.find(s) == name_to_int.end())
    {
        name_to_int[s] = m_count;
        schools.emplace_back();
    }
    schools[name_to_int[s]].push_back(i+1);
}
```

In Python, you can use a `dict` to associate the names of schools with integers.

```python
schools = dict()
for i,s in enumerate(schools):
    if s not in schools:
        schools[s] = []
    schools[s].append(i+1)
```

Subtask 1

Subtask 1 is doable using brute-force. Try all possible $n!$ arrangements of students and check if any of them are safe. This is doable in $O(n! \cdot n)$ time (since it takes $O(n)$ to check an arrangement.)

Subtask 2

In solving this problem, we prove a necessary and sufficient condition for a safe seating arrangement to be possible.

$$\boxed{\text{A safe seating arrangement is impossible if and only if more than } \left\lceil \frac{n}{2} \right\rceil \text{ students hail from the same school.}}$$

($\Leftarrow$) Without loss of generality, suppose $\left\lceil \frac{n}{2} \right\rceil + 1$ students hail from school 1. In any safe seating arrangement, there must be at least one student from a different school between successive students from school 1. So the total number of students is at least

$$\begin{pmatrix}\text{number of students}\\ \text{from school 1}\end{pmatrix} + \begin{pmatrix}\text{number of students}\\ \text{from other schools}\end{pmatrix} = \left(\left\lceil \frac{n}{2} \right\rceil + 1\right) + \left(\left\lceil \frac{n}{2} \right\rceil\right) = 2\left\lceil \frac{n}{2} \right\rceil + 1 > n.$$

It's impossible for a seating arrangement to have more than n students, hence a safe seating arrangement is impossible.

($\Rightarrow$) To show that "If a safe seating arrangement is impossible, then more than $\left\lceil \frac{n}{2} \right\rceil$ students hail from the same school.", we prove the contrapositive: "If at most $\left\lceil \frac{n}{2} \right\rceil$ students hail from the same school, then a safe seating arrangement is possible.".

This can be proved by construction, which incidentally, is exactly the programming problem we are to solve.

Suppose there are only two schools.

Without loss of generality, suppose for a given input that there are at most $M \le \lceil \frac{n}{2} \rceil$ students from the same school, and that there are M students hailing from school 1.

How many students are there from school 2? There are $n - M \ge \lfloor \frac{n}{2} \rfloor$ students from school 2.

We assumed $M \le \left\lceil \frac{n}{2} \right\rceil$, so $-M \ge -\left\lceil \frac{n}{2} \right\rceil$. Since $\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor = n$, we have

$$\begin{aligned} n &= \left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor \\ n - M &\ge \left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor - \left\lceil \frac{n}{2} \right\rceil \\ n - M &\ge \left\lfloor \frac{n}{2} \right\rfloor . \end{aligned}$$

So there are at least $\lfloor \frac{n}{2} \rfloor$ students from school 2. Since $M \le \lceil \frac{n}{2} \rceil$, it follows that $M - 1 \le \lfloor \frac{n}{2} \rfloor$. So there are at least $M - 1$ students from school 2.

So actually, there are only two possible cases:

- Both schools contain the same number of students (M).
- One school contains M students and the other school contains $M - 1$ students.

In either case, you can form a safe seating arrangement by alternating students from each school.

Subtasks 3 and 4

What if there are more than two schools?

So if we know all schools have at most $\left\lceil \frac{n}{2} \right\rceil$ students, how do we construct a safe seating arrangement? Without loss of generality, suppose for a given input that there are at most $M \le \left\lceil \frac{n}{2} \right\rceil$ students from the same school, and that there are M students hailing from school 1.

Remark. You do not need to sort anything in order to identify the most populous school. Simply check the length of `schools[i]` for all `i` and record the `i` which gives the longest length. This can be done in $O(m)$ time.

Consider the $M - 1$ gaps between students from school 1, as well as the possibility of placing students after the last student from school 1. This gives us M slots we can put other students in.

1 _ 1 _ 1 _ 1 _ ... _ 1 _

All slots (except possibly the last one) must have at least one student from one of the other schools. This is possible since there are $n - M \ge \left\lfloor \frac{n}{2} \right\rfloor \ge M - 1$ students from other schools.

Furthermore, since all other schools have at most M students, we can additionally enforce that a slot cannot have two people from the same school.

To actually construct a solution, we successively assign students to slots, one school at a time. Arrange all of the other students, so in order, we have students from school 2, then school 3, and so on. We can go through the students in this arrangment, assigning one student to each slot, returning to the first slot after we fill in the last slot.

Example. Suppose there are five students from school 1, two students from school 2, four students from school 3, and five students from school 4. Initially, we have the following empty slots:

1 _ 1 _ 1 _ 1 _ 1 _

The students we have to put in the slots are in the order $2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4$.

1 _ 1 _ 1 _ 1 _ 1 _	students left : $2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 4$
1 **2** 1 _ 1 _ 1 _ 1 _	students left : $2, 3, 3, 3, 3, 4, 4, 4, 4, 4$
1 2 1 **2** 1 _ 1 _ 1 _	students left : $3, 3, 3, 3, 4, 4, 4, 4, 4$
1 2 1 2 1 **3** 1 _ 1 _	students left : $3, 3, 3, 4, 4, 4, 4, 4$
1 2 1 2 1 3 1 **3** 1 _	students left : $3, 3, 4, 4, 4, 4, 4$
1 2 1 2 1 3 1 3 1 **3**	students left : $3, 4, 4, 4, 4, 4$
1 2 **3** 1 2 1 3 1 3 1 3	students left : $4, 4, 4, 4, 4$
1 2 3 1 2 **4** 1 3 1 3 1 3	students left : $4, 4, 4, 4$
1 2 3 1 2 4 1 3 **4** 1 3 1 3	students left : $4, 4, 4$
1 2 3 1 2 4 1 3 4 1 3 **4** 1 3	students left : $4, 4$
1 2 3 1 2 4 1 3 4 1 3 4 1 3 **4**	students left : 4
1 2 3 **4** 1 2 4 1 3 4 1 3 4 1 3 4	students left : none

We will always fill up the first $M - 1$ slots since there are at least $M - 1$ students from other schools. Additionally, we will never place two students from the same school in the same slot, since there are at most M students from the same school. So this algorithm will always give us a safe seating arrangement.

This can be implemented in $O(n)$ time. Using the `schools` table to act as our "students left" pool, we can transfer students into M buckets/slots according to the algorithm above in $O(n)$. Then, it takes $O(n)$ to go through each of the buckets and piece together the final arrangement. No sorting is required.

A suboptimal solution may run in $O(nm)$, which would pass subtask 3 but not 4.