2.1 Wedding Shuffle

A number of wedding guests are seated at a circular table. The master of ceremonies asks all the guests to stand, and sit back down either one seat to the left, one seat to the right, or in the same chair.

All guests must have a chair once everyone is seated.

Your task is to determine all possible ways that the guests can do this "wedding shuffle."

2.3 Wedding Shuffle with Barriers

This second problem is similar, but now the table has a set of plexiglass barriers in between some of the chairs.

Write a program `wedding.cpp` with a class `Wedding` which has several methods as shown below:

```
class Wedding {
  public:
    Wedding();
    vector<string> shuffle(string guests);
    vector<string> shuffle_barriers(string guests, vector<int> barriers);

};
```

You may add other methods and data structures to the class as necessary to solve the problem.

The Wedding Class

Each `Wedding` object uses a fixed set of names for the guests. The guests are labelled with individual characters in a string. So, for example, the string

`abc12`

represents five guests whose names are `a`, `b`,`c`, `1`, and `2`.

The `shuffle` method

The `shuffle` method should return all possible arrangements of the guests specified as a vector of strings. The initial seating arrangement is as specified in the parameter `guests`. For visualization, suppose that `guests[0]` is seated at the one-o'clock position of a clock face, and `guests[1]` is to their left (at 2 oclock). `guests[n]` is to the left of `guests[n-1]` until we arrive back to `guests[0]` The last guest is to the immediate right of `guests[0]`, they are seated at "11 o'clock."

Here is a simple example. If the guests are `abcde`, then the result of `shuffle("abc")` will be

```
abc
acb
bac
bca
cab
cba
```

The `shuffle_barriers` method

The `shuffle_barriers` method should return all possible arrangements of guests as a vector of strings.

The barriers are specified as a vector of integers. A barrier always comes before the position it specifies. So a barrier at 0 means that the person seated at position 0 cannot move to their right (to position n-1). Using the clock face analogy, a barrier at 0 is like a barrier at 12-o'clock (just to the right of `guests[0]`)

The barriers should be included in the output as a visual aid, using the `|` character.

Here is a simple example: the result of `shuffle_barriers("abcd",{2})` will be

```
ab|cd
ab|dc
ba|cd
ba|dc
db|ca
```

Program name

The template program and the `main` to be used for testing is provided in wedding_original.cpp

Libraries allowed

You must include the following libraries

- `<algorithm>`
- `<iostream>`
- `<map>`
- `<string>`
- `<vector>`

For the moment, no other includes are permitted.

No brackets

Brackets (i.e. `[]`) are not permitted. Use the access and modifier methods instead.