(a) You will write the `Flight` member function `EmptySeatCount`, which is described as follows. `EmptySeatCount` returns the number of empty seats of the specified type `seatType`. Recall that an empty seat holds a default passenger whose name is `""`. If `seatType` is `"any"`, then every empty seat should be counted in determining the number of empty seats. Otherwise, only seats whose type is the same as `seatType` are counted in determining the number of empty seats.

For example, consider the diagram of passengers assigned to seats as stored in `mySeats` for `Flight ap2002` as shown below.

	[0]	[1]	[2]	[3]	[4]	[5]
[0]	window "Kelly"	middle "Robin"	aisle ""	aisle "Sandy"	middle ""	window "Fran"
[1]	window "Chris"	middle "Alex"	aisle ""	aisle ""	middle "Pat"	window "Sam"

The following table shows several examples of calling `EmptySeatCount` for this flight.

Function Call	Value Returned
`ap2002.EmptySeatCount("aisle")`	`3`
`ap2002.EmptySeatCount("window")`	`0`
`ap2002.EmptySeatCount("middle")`	`1`
`ap2002.EmptySeatCount("any")`	`4`

Complete function `EmptySeatCount` below.

```
int Flight::EmptySeatCount(const apstring & seatType) const
// postcondition: returns the number of empty seats
//                whose type is seatType;
//                if seatType is "any", returns the
//                total number of empty seats
```

(b) You will write the `Flight` member function `FindBlock`, which is described as follows. `FindBlock` searches for a block of `seatsNeeded` adjacent empty seats in the specified row. If such a block of seats is found, `FindBlock` returns the column index of the first (i.e., the lowest index) seat in the block; otherwise, it returns -1.

The seating diagram for passengers of `Flight ap2002` is repeated here for your convenience.

	[0]	[1]	[2]	[3]	[4]	[5]
	window	middle	aisle	aisle	middle	window
[0]	"Kelly"	"Robin"	""	"Sandy"	""	"Fran"
	window	middle	aisle	aisle	middle	window
[1]	"Chris"	"Alex"	""	""	"Pat"	"Sam"

The following table shows several examples of calling `FindBlock` for `Flight ap2002 as` shown.

Function Call	Value Returned
`ap2002.FindBlock(0, 1)`	`2` or `4`
`ap2002.FindBlock(0, 2)`	`-1`
`ap2002.FindBlock(1, 2)`	`2`

Complete function `FindBlock` below.

```
int Flight::FindBlock(int row, int seatsNeeded) const
// postcondition: returns column index of the first (lowest index)
//                seat in a block of seatsNeeded adjacent
//                empty seats in the specified row;
//                if no such block exists, returns -1
```

(c) You will write the `Flight` member function `AssignGroup`, which is described as follows. The parameter to the `Flight` member function `AssignGroup` is an array of passengers, `group`. These passengers require a block of adjacent seats in a single row. `AssignGroup` searches for `group.length()` adjacent empty seats in some row. If such a block of seats is found, the passengers in `group` will be assigned to those seats, and `AssignGroup` returns `true`. Otherwise, no passengers are assigned to seats, and `AssignGroup` returns `false`.

For example, the seats in `Flight ap314` are as shown in the first diagram below. If the array `adults` contains three passengers, the call `ap314.AssignGroup(adults)` makes no changes to `ap314` and returns `false`, because there is no block of three adjacent empty seats in a single row. On the other hand, suppose the array `kids` contains passengers `"Sam"` and `"Alex"`. The call `ap314.AssignGroup(kids)` will assign `"Sam"` and `"Alex"` to the seats shown in the second diagram below and return `true`.

Contents of `mySeats` for `ap314` before any call to `AssignGroup`

	[0]	[1]	[2]	[3]	[4]
[0]	window "Kelly"	aisle ""	aisle "Sandy"	middle ""	window "Fran"
[1]	window "Chris"	aisle ""	aisle ""	middle "Pat"	window ""

Contents of `mySeats` for `ap314` after call to `ap314.AssignGroup(kids)`

	[0]	[1]	[2]	[3]	[4]
[0]	window "Kelly"	aisle ""	aisle "Sandy"	middle ""	window "Fran"
[1]	window "Chris"	aisle "Sam"	aisle "Alex"	middle "Pat"	window ""

In writing `AssignGroup`, you may call `FindBlock` specified in part (b). Assume that `FindBlock` works as specified, regardless of what you wrote in part (b).

Complete function `AssignGroup` below.

```
bool Flight::AssignGroup(const apvector<Passenger> & group)
// postcondition: if possible, assigns the group.length() passengers
//                from group to adjacent empty seats in a single row
//                and returns true;
//                otherwise, makes no changes and returns false
```

END OF EXAMINATION