4. Consider the problem of assigning passengers to seats on airline flights. Three types of information are needed —passenger information, seat information, and flight information. Three classes will be used to represent this information, respectively: `Passenger`, `Seat`, and `Flight`.

You will write three member functions for the `Flight` class:

(a) `EmptySeatCount` that returns the number of empty seats of a specified type,

(b) `FindBlock` that returns information about the location of an empty block of seats, and

(c) `AssignGroup` that attempts to assign a group of passengers to adjacent seats.

Passenger information is abstracted by a class and includes a name and other information. A default passenger, used to indicate “no passenger” in a seat, has the empty string `""` as its name. The declaration for class `Passenger` is as follows.

```
class Passenger
{
  public:
    Passenger();     // default passenger with name ""

    apstring GetName() const;
    // postcondition: returns passenger’s name

    // ... other public and private members not shown
};
```

Seat information includes the passenger assigned to the seat and the type of the seat (“window”, “aisle”, “middle”). The `Seat` function `GetPassenger` returns the passenger assigned to the seat; if the seat is empty, `GetPassenger` returns a default passenger. The declaration for the class `Seat` is as follows.

```
class Seat
{
  public:
    Passenger GetPassenger() const;
    // postcondition: returns passenger in this seat

    apstring GetType() const;
    // postcondition: returns the type of this seat

    void SetPassenger(const Passenger & p);
    // postcondition: assigns p to this seat (i.e., GetPassenger() == p)

    // ... constructors and other public and private members not shown
};
```

GO ON TO THE NEXT PAGE.

Seat assignments are processed by the public member functions of the class Flight. The seating arrangement is represented internally by a matrix of seats in the class Flight. The declaration for the class Flight is as follows.

```
class Flight
{
  public:
    int 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

    int 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

    bool 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

    // ... constructors and other public member functions not shown

  private:
    apmatrix<Seat> mySeats;

    // ... other private data members not shown
};
```