The materials included in these files are intended for use by AP teachers for course and exam preparation in the classroom; permission for any other use must be sought from the Advanced Placement Program®. Teachers may reproduce them, in whole or in part, in limited quantities, for face-to-face teaching purposes but may not mass distribute the materials, electronically or otherwise. These materials and any copies made of them may not be resold, and the copyright notices must be retained as they appear here. This permission does not apply to any third-party copyrights contained herein.
Complete function `EmptySeatCount` below.

```cpp
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
```

```cpp
int row, col, counter = 0;
int rowlen = mySeats.rows(), collen = mySeats.cols();
for (row = 0; row < rowlen; row++)
    for (col = 0; col < collen; col++)
        if (mySeats[row][col].GetPassenger().GetName() == "")
            if (mySeats[row][col].GetType() == seatType ||
                mySeats[row][col].GetType() == "any")
                counter++;

return counter;
```

GO ON TO THE NEXT PAGE.
Complete function `FindBlock` below.

```cpp
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

int col, collen = mySeats.size(), counter = 0;
for (col = 0; col < collen; col++)
    if (mySeats[row][col].GetPassenger().GetName() == """)
        counter++;
    else
        counter = 0;
    if (counter == seatsNeeded)
        return col - seatsNeeded + 1;

return -1;
```

GO ON TO THE NEXT PAGE.
Complete function AssignGroup below.

```cpp
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

    int i, row, rowlen = mySeats.numRows(), groupLen = group.length(), temp;
    for (row = 0; row < rowlen; row++) {
        temp = FindBlock(row, groupLen);
        if (temp != -1) {
            for (i = 0; i < groupLen; ++i)
                mySeats[row][temp + i].setPassenger(group[i]);
            return true;
        }
    }

    return false;
}
```

GO ON TO THE NEXT PAGE.
Complete function `EmptySeatCount` below.

```c++
int Flight::EmptySeatCount(const astring & 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 i, j, num=0;
    for( i = 0; i < mySeats.numrows(); i++ )
        for( j = 0; j < mySeats.numcols(); j++ )
            if (mySeats[i][j] . getType == seatType || seatType == "any")
                if (mySeats[i][j] . getPassenger().getName() == "")
                    num++;
    return num;
}
```

GO ON TO THE NEXT PAGE.
Complete function `FindBlock` below.

```cpp
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
{
    int i, num = 0;
    for (i = 0; i < mySeats.numCols(); i++)
        if (mySeats[row][i].GetPassenger().GetName() == "")
            num ++;
        if (num == seatsNeeded)
            return i - seatsNeeded + 1;
    else
        num = 0;
    return -1;
}
```

GO ON TO THE NEXT PAGE.
Complete function AssignGroup below.

```cpp
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

    int i, pos, x = group.length();

    for (i = 0; i < mySeats.numrow(); i++) {
        pos = Find Block (i, x);
        if (pos != -1)
            break;
    }

    if (pos == -1)
        return false;

    else
        for (q = 0; q < x; q++)
            mySeats [i] [pos + q].set Passenger (group[i]);

    return true;
}
```

GO ON TO THE NEXT PAGE.
Complete function `EmptySeatCount` below.

```c
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
{
    int empty = 0;
    for (int j = 0; j < mySeats.numrows(); j++)
    {
        for (int k = 0; k < mySeats.numcols(); k++)
        {
            if (mySeats[j,k].GetType() == seatType || seatType == "any")
                if (mySeats[j,k].GetName() == "")
                { empty = empty + 1; }
        }
    }
    return empty;
}
```

GO ON TO THE NEXT PAGE.
Complete function `FindBlock` below.

```cpp
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

    for (int k = 0; k < numCols(); k++)
    
        if (mySeats[row][k].getName() == "")
        
            for (int j = 0; j < seatsNeeded; j++)
            
                if (mySeats[row][k+j].getName() == "") &
                    (j == seatsNeeded)
                    return k;

            else
                j = seatsNeeded

        return -1;

GO ON TO THE NEXT PAGE.
Complete function AssignGroup below.

```cpp
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

    int seatsNeeded, BlockStart, i;
    seatsNeeded = group.length();
    i = 0;
    for (int k = 0; k < mySeats.numRows(); k++)
    {
        BlockStart = FindBlock(k, seatsNeeded);
        if (BlockStart > 0)
        {
            for (int j = BlockStart; j < seatsNeeded; j++)
            {
                mySeats[k][j].SetPassenger(group[i]);
                i += 1;
            }
            return true;
        }
    }
    return false;
}
```

GO ON TO THE NEXT PAGE.