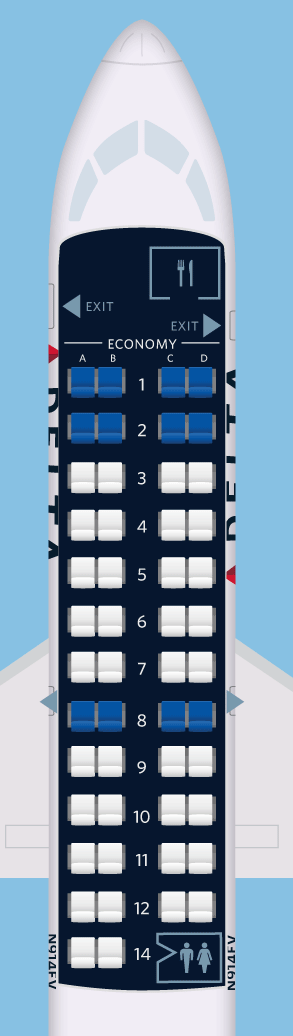
ECCS 1611 – Programming 1

Lab 9 – More Arrays and Vectors

**Part A. Smxall Group Problem Solving (you can discuss this with your teams or others at lab) + Individual Coding**

The goal of this assignment is to write a program to assign passengers seats in an airplane. We will assume that a small airplane, such as the Canadair CRJ flown by many regional airlines, is the one being used.

As shown in the figure to the left (courtesy of seatguru.com), the CRJ has a total of 50 seats, configured as 12 rows of 4 seats (with an aisle between seats B and C) and a thirteenth row (but labeled as row 14 – perhaps for triskaidekaphobic customers).

The seating should be initially shown as follows:

1 A B C D

2 A B C D

3 A B C D

4 A B C D

5 A B C D

6 A B C D

7 A B C D

8 A B C D

9 A B C D

10 A B C D

11 A B C D

12 A B C D

14 A B

The program should display the seat pattern as shown above, with an ‘X’ marking the seats already assigned. For example, after seats 1A, 2B and 4C are assigned, the first four rows of the display should look like this:

1 X B C D

2 A X C D

3 A B C D

4 A B X D

The program is to operate as follows. After displaying the seats both available and occupied, the program allows the user to enter in a seat (you may assume the input is always correctly formatted as an integer followed by a character); the display is then updated. This continues until either all seats are filled or the user signals that the program should end. If the user types in a seat that is already assigned, the program is to indicate that the seat is occupied and to then prompt for another choice. Similarly, if the seat does not exist, then inform the user accordingly. For your programming convenience, the seating chart must be implemented using a **GLOBAL** 2-D character array (which means that you do not have to pass the array to functions – it is already accessible).

***BEFORE diving in to start coding the solution to this problem, discuss and answer the questions provided on your Lab 11 Part A Checksheet.***

**Part B. Individual Programming Practice**

**P9-1** Credit Card Number Check, part 1 (revisiting from pre-lab 5 / lab 5). The last digit of a credit card number is the check digit, which protects against transcription errors such as an error in a single digit or switching two digits. The following method is used to verify actual credit card numbers but, for simplicity, we will describe it for numbers with 8 digits instead of 16:

• Starting from the rightmost digit, form the sum of every other digit. For example, if the credit card number is 43589795, then you form the sum: 5 + 7 + 8 + 3 = 23.

• Double each of the digits that were not included in the preceding step. Add all of the individual digits of the resulting numbers. For example, with the number given above, the digits selected are:

9 9 5 4

Doubling these digits yields:

18 18 10 8

Adding all digits in these values yields:

(1 + 8) + (1 + 8) + (1 + 0) + (0 + 8) = 27.

• Add the sums of the two preceding steps. If the last digit of the result is 0, the number is valid. In our case, 23 + 27 = 50, so the number is valid.

Write a program that implements this algorithm, using the function prototype:

**bool isCardValid( int digits[], int size );**

This function is passed an integer array that ***separately*** stores each digit of the credit card number and returns true only if the 8-digit credit card number is valid. The user should supply an 8-digit number, which is to be read in as a string, and the program should indicate whether or not the card number is valid.

Example run (with user input indicated with ***bold italics***):

Enter 8-digit credit card # or Q to quit: ***43589795***

Card is valid.

Enter 8-digit credit card # or Q to quit: ***43589796***

Card is not valid.

Enter 8-digit credit card # or Q to quit: ***Q***

**P9-2** Credit Card Number Check, part 2: revise the program written in P11-1 such that it works with 16-digit credit card numbers.

Example runs (with user input indicated with ***bold italics***):

Enter 16-digit credit card # or Q to quit: ***4010046804271206***

Card is valid.

Enter 16-digit credit card # or Q to quit: ***4010046804271208***

Card is not valid.

Enter 16-digit credit card # or Q to quit: ***Q***

**P9-3** Write the function append (using the prototype shown below) that appends one vector after another as part of a program that performs appropriate testing of this function, including a helper function for displaying the contents of a vector. For example, if a contains:

1 4 9 16

and b contains:

9 7 4 9 11

then append returns a vector containing:

1 4 9 16 9 7 4 9 11

Function Prototype: vector<int> append(vector<int> a, vector<int> b);