CMPE 50 – Spring 2015, Tarng
Lab #7– Pointer and Dynamic Arrays

Instructions: Before you leave the lab, you should submit your answers through Canvas->Assignment->Lab7->Submission. Please submit your answers (.cpp files, and output files with appropriate documentation/comments) even if you couldn't complete/run them.

You only need to work on Exercise 1 in the lab. Exercise 2 is optional.

Exercise 1:
(Based on Programming Projects 9.3 and 7.11)
Write a program to assign passengers seats in an airplane. Assume a small airplane with seat numbering 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
```

Your program will ask the user how many rows the plane has and use a dynamic array or arrays) to handle the number of rows. Your program should display the seat pattern, with an X marking the seats already assigned. For example, after seats 1A, 2B, and 4C are taken, 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
5  A B C D
6  A B C D
7  A B C D
```

After displaying the seats available, the program prompts for the seat desired, the user types in a seat, and then the display of available seats is updated. This continues until all seats are filled or until the user signals that the program should end. If the user types in a seat that is already assigned, the program should say that the seat is occupied and ask for another choice. Don't forget to free the memory allocated for these returned dynamic arrays when the data is no longer needed.

(Automated testing) For a large number of rows, you may need to enter a lot of test data. In order to speed up the testing, you need to design your program to allow the input to be entered via console or an input file. So you should initially prompt the user to ask whether the input will come from the console or an input file. If from an input file, user needs to enter the file name. For example, the test file should contain the following input:

```
7
1A
2A
3A
4A
5A
6A
7A
2B
2C
2D
…
7D
end
```

You can have different test files, with some containing conflicting seat assignments.

(Hint: To make your program automatically testable, replace the cin with an istream variable so you can control where the input stream come from in the main program.)

(More hint:
There are two approaches to define the arrays. The first approach is to define four arrays, each represent a column of seats. See the following code fragment:

```
char *colA;
char *colB;
char *colC;
char *colD;

cin >> rowSize;

colA = new char[rowSize];
colB = new char[rowSize];
colC = new char[rowSize];
colD = new char[rowSize];
```

The second approach is to use a two-dimensional array to represent the seats, as follows:

```
char **seats;

cin >> rowSize;

seats = new char*[rowSize];

for (int i = 0; i < rowSize; i++)
{
     seats[i] = new char[4];
}
```

(Optional) Exercise 2:
(Based on Programming Projects 9.2 and 7.2)
Write a function called `delete_repeats` that has a partially filled array of characters as a formal parameters and that deletes all repeated letters from the array. It returns a new dynamic array where all repeated letters are deleted. Do not modify the input array.

For example, consider the following code:

```
char str[100] = “to be or not to be”;
int size = strlen(str);
char *noRepeat;
noRepeat = delete_repeats(str, size);

cout <<  noRepeat;
```

Will print out

```
to bern
```

Let’s assume that the characters are case insensitive. Don’t forget to free the memory allocated for these returned dynamic arrays when the data is no longer needed.