Fall 2016 CST8234 – C Programming
Lab 03: Arrays

Setup:

1. Create a directory `Lastname_03` you are going to develop your lab here!
2. Copy `03_100_Skeleton.c` and rename it as `03_100.c`

Program #1:

There are 100 doors in a long hallway. They are all closed. The first time you walk by each door, you open it. The second time around, you close every second door (since they are all opened). On the third pass you stop at every third door and open it if it's closed, close it if it's open. On the fourth pass, you take action on every fourth door. You repeat this pattern for 100 passes.

Question: At the end of 100 passes, what doors are opened and what doors are closed?

Write a small C program: `02_100.c` that simulates this behaviour.

The following demonstrates the execution of the program:

```
# ./02_100
  1  0  0  1  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  0  0  1
# echo $?
0

                          SAMPLE TEST OUTPUT:  02_100
```

In order to successfully complete this program and obtain all the marks, you will need to:

1. Define a `SIZE` macro in your program with a value of 100. Use `SIZE` define the size of your array, and in all your `for` loops
2. Define a *local array to* `main( )` of size `SIZE` to represent the state of the doors. Initialize to 0s all location, using the following declaration:

```
      char array[ SIZE ] = { 0 };
```

3. Write a funtion, with function prototype (exaclty written) `int toggle_door( char a[ ] )`
   (A) `toggle_door( )` should make a 100 passes throug the doors
   (B) If the door is open, it should close it
   (C) If the door is closed, it should open it
   (D) `toggle_door( )` should return 0 to `main( )`
4. Write a function with function prototype (exactly written) `int doors_state( char a[ ] )`
   (A) Displays the status of the doors. Simple print of the array – values 0s and 1s
   (B) `doors_state( )` returns to `main( )` with value 0
5. Your `main( )` function should:
   (A) Call `toggle_door()`
   (B) Call `doors_state( )`
   (C) Terminates with a `return` statement with value `EXIT_SUCCESS`

6. Check the exit status of your program from the command line `echo $?` should display a successful exit
7. Your program should be compiled with the flags `-Wall -ansi -pedantic`

Program #2:

Copy your `02_100.c` to a new program `02_100_B.c` and modify your function `doors_state( )` to print a table instead of the complete array in one line. Use the function `printf( )`, but do not use a TAB. Instead use a field modifier to specify the width of the number you want to print. Use a width of 3 or 4.

In the output shown below, a table of 10 rows is used. The row and column number are displayed to make finding the positions of the array easier to the user. Position #49 is shown as o (Open).

No script is given to test.

```
# ./02_100_B
   0    1    2    3    4    5    6    7    8    9   10
---------------------------------------------
   0    O    C    C    O    C    C    C    C    O    C
  10    C    C    C    C    C    O    C    C    C    C
  20    C    C    C    C    O    C    C    C    C    C
  30    C    C    C    C    C    O    C    C    C    C
  40    C    C    C    C    C    C    C    C    O    C
  50    C    C    C    C    C    C    C    C    C    C
  60    C    C    C    O    C    C    C    C    C    C
  70    C    C    C    C    C    C    C    C    C    C
  80    O    C    C    C    C    C    C    C    C    C
  90    C    C    C    C    C    C    C    C    C    O
```

SAMPLE TEST OUTPUT: 02_100_B