COS 135 Final Project (20%)

Due: Friday 05/05/23 End of the day

[100pts] Write a C program for the following task and submit your source code (you must submit .c file without compilation errors). **DO NOT MODIFY THE GIVEN DATASET (i.e., carpark.txt).**

This project may be completed alone or in pairs (maximum of two students per team). The final submission should include an interactive C program. Please see below for the project brief and deliverables. If you work in pairs, only one member should submit the final materials to Brightspace.

What to submit:

- Please submit a .c source file.

[-5 pts each] Source codes should be able to compile and execute without errors or warnings:

- Source codes should be able to compile without any errors
- Always compile your source codes with -Wall (enable all warnings) switch to find all the compiler warnings and fix them (see example below).

```
$ gcc mycode.c  -o  myprogram -Wall
```

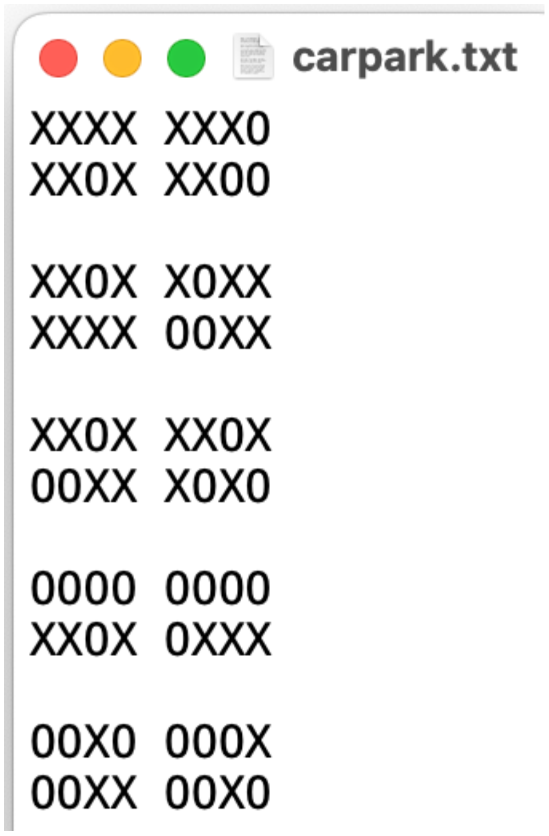
[-10 pts each] **Comments** are required in the following locations (in each C source code):

- **[-2 pts]** At the top of the source code, comment your name (both names if you work in pairs), each of your contributions, and insert a short program description.
- **[-4 pts]** Comment the purpose of each variable.
- **[-4 pts]** Comment major sections of your code such as inputs, functions, and outputs.

Program Design: Your program is a professional document and must be neat and easy to read. All programs should follow these specifications.

- Comments should be aligned and entered in a consistent fashion
- Blank lines should be added to aid readability
- Code within blocks should be indented
- Comments should not contain spelling mistakes
- Variable names should be meaningful
- Define functions where necessary
- Take time to design your code (program flow and functions) before start coding

An automated car park (parking lot) is controlled using a sensor platform. The sensor platform updates **carpark.txt** with the latest status of the carpark. Text file **carpark.txt** contains the layout of the car park with 14 rows and 9 columns (which includes 4 empty rows and 1 empty column). Available slots are marked with the character ‘O’ (uppercase character O) and occupied slots are marked with character ‘X’.



```
XXXX XXX0
XX0X XX00

XX0X X0XX
XXXX 00XX

XX0X XX0X
00XX X0X0

0000 0000
XX0X 0XXX

00X0 000X
00XX 00X0
```

Design and develop a C program to remotely book a free slot in the car park using the Ubuntu terminal. First, **the data should be read from the carpark.txt file into an appropriate C data structure (array).**

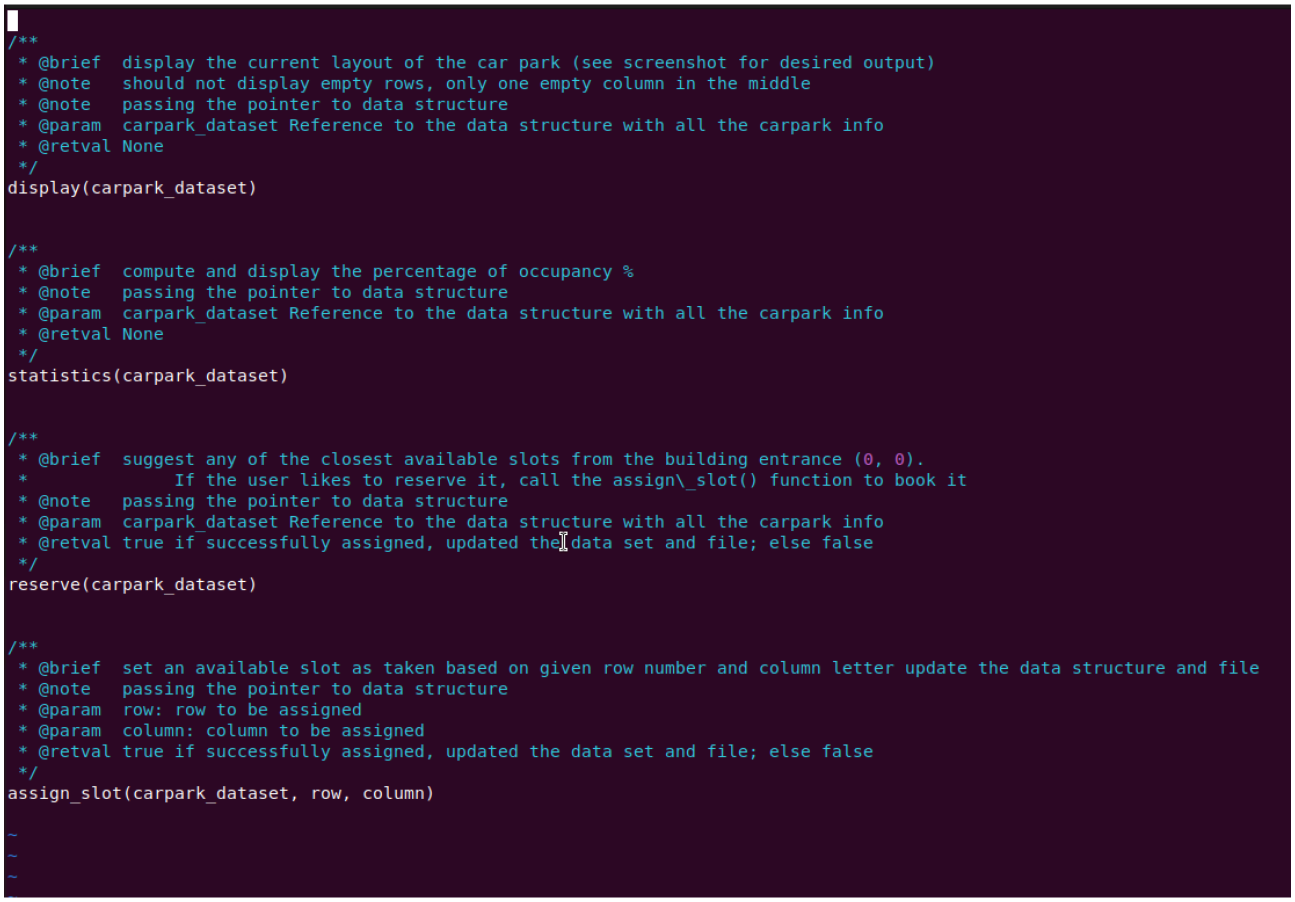
Pro tip: it will be helpful to prepare a flow chart of the program (considering all its functionalities and features) before you start coding. Remember – design first!

Possible error conditions should be appropriately handled with error messages and help texts.

<u>Getting started (sample skeleton of the code):</u>

Your program should implement the following functions.

(showing only pseudo functions, alter the function parameters and define additional functions as required)



```
/**
 * @brief  display the current layout of the car park (see screenshot for desired output)
 * @note   should not display empty rows, only one empty column in the middle
 * @note   passing the pointer to data structure
 * @param  carpark_dataset Reference to the data structure with all the carpark info
 * @retval None
 */
display(carpark_dataset)


/**
 * @brief  compute and display the percentage of occupancy %
 * @note   passing the pointer to data structure
 * @param  carpark_dataset Reference to the data structure with all the carpark info
 * @retval None
 */
statistics(carpark_dataset)


/**
 * @brief  suggest any of the closest available slots from the building entrance (0, 0).
 *              If the user likes to reserve it, call the assign\_slot() function to book it
 * @note   passing the pointer to data structure
 * @param  carpark_dataset Reference to the data structure with all the carpark info
 * @retval true if successfully assigned, updated the data set and file; else false
 */
reserve(carpark_dataset)


/**
 * @brief  set an available slot as taken based on given row number and column letter update the data structure and file
 * @note   passing the pointer to data structure
 * @param  row: row to be assigned
 * @param  column: column to be assigned
 * @retval true if successfully assigned, updated the data set and file; else false
 */
assign_slot(carpark_dataset, row, column)

~
~
~
```

Tasks:

1. **[20pts]** There should be a repeatable text-based menu with help text as below:

```
Select your choice from the menu:
D to display current status of the car park
R to reserve an empty slot in the car park
S to display % occupancy of the car park
Q to quit
:
```

After the colon (:), a user can enter the menu choice character (D, R, S, or Q) and press enter.

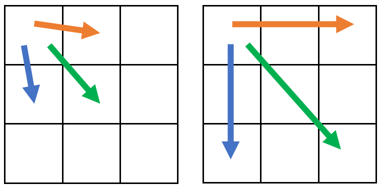
2. **[10pts]** The program should **quit** only if a user enters Q.

3. **[20pts]** The **display** option should: 1) read the data from the text file, and 2) display the data (display row numbers down the left side and column letters - A, B, C, D, E, F, G, and H - across the top line).

- There should be a blank column similar to the text file indicating the road but no blank rows

4. **[20pts]** The **reserve** function should suggest any of the closest available slots from the building entrance (assume that the building entrance is at the top left corner where row = 0 and column = 0).

   a) Users can also provide a row number and column letter to reserve an empty slot.
   b) If the data entered (row and column) is out of range, report an error message and the user can try again next pass of the menu loop.
   c) If the seat is not available report an error message.
   d) If the seat is available report so and update the data structure and text file (replace O with X).



To find the closest available slot, you should ***develop a nearest neighbor searching algorithm*** as illustrated.
(start searching for an empty slot in neighbors from the top left)

Nearest Neighbor Searching: develop an algorithm to search in terms of matrices, first, 2x2 (top left of the data set), then 3x3, followed by 4x4 and continue until 8x8. If no free slot can be found before 8x8 stop searching and ask for manual booking.

5. **[10pts]** The **statistics** function should report the % occupancy of the carpark. This is computed as the number of occupied slots divided by total slots. Set precision to 1 decimal.

occupancy % = (number of occupied slots / total slots) * 100

NOTE: sample program execution and outputs are displayed at the end.

Sample program execution and outputs

```
Select your choice from the menu:
D to display current status of the carpark
R to reserve an empty slot in the carpark
S to display % occupancy of the carpark
Q to quit
: D

    ABCD  EFGH
1   XXXX  XXX0
2   XXOX  XX00
3   XX0X  X0XX
4   XXXX  00XX
5   XX0X  XX0X
6   00XX  X0X0
7   0000  0000
8   XX0X  0XXX
9   00X0  000X
10 00XX  00X0

Select your choice from the menu:
D to display current status of the carpark
R to reserve an empty slot in the carpark
S to display % occupancy of the carpark
Q to quit
: R

Closest available slot is 2C, do you want to reserve it (y/n)? n
Please enter your preferred empty slot: 3F
Your selection 3F is reserved!

Select your choice from the menu:
D to display current status of the carpark
R to reserve an empty slot in the carpark
S to display % occupancy of the carpark
Q to quit
: R

Closest available slot is 2C, do you want to reserve it (y/n)? y
Your selection 2C is reserved!
```

```
Select your choice from the menu:
D to display current status of the carpark
R to reserve an empty slot in the carpark
S to display % occupancy of the carpark
Q to quit
: S

Current occupancy is XX.X%

Select your choice from the menu:
D to display current status of the carpark
R to reserve an empty slot in the carpark
S to display % occupancy of the carpark
Q to quit
: D

   ABCD  EFGH
1  XXXX  XXX0
2  XXXX  XX00
3  XX0X  XXXX
4  XXXX  00XX
5  XX0X  XX0X
6  00XX  X0X0
7  0000  0000
8  XX0X  0XXX
9  00X0  000X
10 00XX  00X0


Select your choice from the menu:
D to display current status of the carpark
R to reserve an empty slot in the carpark
S to display % occupancy of the carpark
Q to quit
:
```

* Note that the 3F and 2C slots are now occupied in the dataset. To achieve this, you need to write the updated dataset to the carpark.txt file every time you update the dataset. Please do not change the format of the dataset in carpark.txt when you append data.