CC213 Programming Application Project Ideas:

Responsible use of online resources:
Please cite the web sites that you download examples and resources from. Copying partial contents from the Internet is alright to learn from. If you submit fully copied projects, you will receive a zero mark. Please add more features, or change the behaviour of the function, or the interface at least. Please remember that you will display a demo of your project running and explain how you wrote the source code. If the project doesn't run, you will loose a lot of marks. Similarly, if you fail to explain how the source code was written, you will loose marks.

General Rules:
- A group can be formed from 4 or 5 students per project.
- Groups are not allowed to repeat ideas, so first come first serve bases, once a group comment on the project's thread with a choice, the idea can not be chosen by another group.

Grading Criteria:
The project is worth10% of your final mark. These marks are graded individually by asking each student in the presentation for their contribution and testing their understanding. These are broken into:
- 5 marks for correctness (no compilation or run-time errors).
- 4 marks for the application of course concepts, where feasible, (such as: 2D arrays, pointers, pass by reference, strings, structs and unions, recursion, Text and binary files, bitwise Operators and the studied applications)
- 1 marks for the interface, presentation, documentation, and teamwork.
- 5 bonus marks for all other concepts you self-study outside the learning objectives of the course.

Submission Details:
All project files are zipped and submitted named as "proj_LeaderStudentID.zip", where "LeaderStudentID" is replaced by a leader team member chosen by all team members. The zip file should contain:
- All source code used to develop the project, either as a jar file, or a folder of the project packages and files.
- A written report (soft-copy and hardcopy) according to the instructions written in the guide and example shown on moodle.

Note:
- Your project will be rejected if it does not comply with the above submission requirements.
- When you submit your project you get 5 marks until you prove that it is your own work, during a demo in Week 15.

1. **Project code: CC213-01:** Tic Tac Toe Game Playing,.

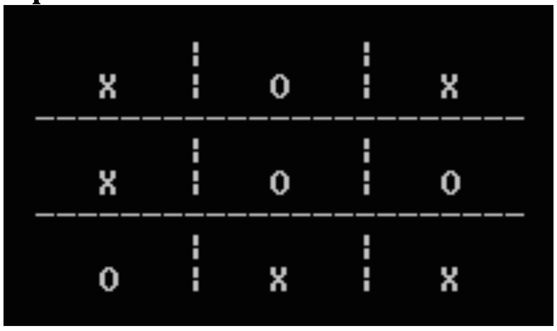
Use a 2D array for the 3x3 grid and 2 players . First Display the grid showing '-' indicating that no characters have been entered. Them ask player one to specify the row index, and column index, in which to place an 'X' character , then redraw the grid. Then ask player two to specify the row index and column index, in which to place an 'O' character , then redraw the grid. Repeat to check a winning situation, or draw situation. Make sure that every time the players enter an empty cell in the grid to enter the new value.

A possible main function outline can look as follows:

```
char tictac[3][3];
int done = 0;
int player = 1;
initialiseBoard(tictac);
while (done == 0) {
        player = (?1: 2; 1);  // toggle players between 1 and 2
        read_position (&row, &col)  // repeat reading until row and col are
between 0:2, and in an empty cell.
        drawBoard (tictac);
        int winner = check_winning(tictac);  // return 1 or 2 if someone won, or 0
otherwise
        if (winner != 0)        // player 1, or 2 won, forming a row, a column or a
diagonal of Xs or Os.
                done = 1;       // then print the winner statement
        done = check_draw(tictac);          // check if the board if full and no one
won.

}
```

A possible draw final board can look as follows:



```
      |     |
   X  |  O  |  X
-------------------
      |     |
   X  |  O  |  O
-------------------
      |     |
   O  |  X  |  X
```

2. **Project code: CC213-02:** Periodic Table:

   Define a structure type element_t to represent one element from the periodic table of elements. Components should include the atomic number (an integer); the name, chemical symbol, and class (strings); a numeric field for the atomic weight; and a seven-element array of integers for the number of electrons in each shell. The following are the components of an element_t structure for sodium.

   11 Sodium Na alkali_metal 22.9898 2 8 1 0 0 0 0

   Define and test I/O functions scan_element and print_element, to read the periodic table from a text file, and print it on the screen.

3. **Project code: CC213-03:** weather and climate research
   You are developing a database of measured meteorological data for use in weather and climate research. Define a structure type measured_data_t with components site_id_num (a four-digit integer), wind_speed, day_of_month, and temperature. Each site measures its data daily, at noon local time. Write a program that inputs a file of measured_data_t records and determines the site with the greatest variation in temperature (defined here as the biggest difference between extrema) and the site with the highest average wind speed for all the days in the file. You may assume that there will be at most ten sites. Test the program on the following July daily data collected over one week at three sites:

ID	**Day**	**Wind Speed (knots)**	**Temperature (deg C)**
2001	10	11	30
2001	11	5	22
2001	12	18	25
2001	13	16	26
2001	14	14	26
2001	15	2	25
2001	16	14	22

3345	10	8	29
3345	11	5	23
3345	12	12	23
3345	13	14	24
3345	14	10	24
3345	15	9	22
3345	16	9	20
3819	10	17	27
3819	11	20	21
3819	12	22	21
3819	13	18	22
3819	14	15	22
3819	15	9	19
3819	16	12	18

4. **Project code: CC213-04:** String Functions, Files I/O:

Write a program that takes words from a text file and prints each one on a separate line of an output file followed by the number of letters (alphabetic characters) in the word. Any leading or trailing punctuation marks shouldbe removed from the word before it is printed. When all the text has been processed, display on the screen a count of the words in the file. Assume that words are groups of non-whitespace characters separated by one or more whitespace characters.

5. **Project code: CC213-05: Airline Reservation System:**
   Develop a small airline reservation system. The database of flight information should be kept in a file of structures with the following components:

   a. Flight number (including airline code)

   b. City of departure

   c. Destination

d. Date and time of departure

e. Date and time of arrival

f. Number of first-class seats still available

g. Number of first-class seats sold

h. Number of coach seats still available

i. Number of coach seats sold

Include in your program separate functions for creation, deletion, and update of flight records. Also, implement make_reservation and cancel_reservation functions, and store valid reservations, in reservations text file.

6. **Project code: CC213-06: Recipes Database:**
Cooking recipes can be stored on a computer and, with the use of files, and can be quickly referenced.

a. Write a function that will create a text file of recipes from information entered at the command line. The format of the data to be stored is:

   i. Recipe type (dessert, meat, poultry, etc.)

   ii. Recipe subtype (for dessert, use cake, pie, or cookies, … etc)

   iii. Recipe name (e.g., German chocolate)

   iv. number of lines in the recipe to follow

   v. the actual recipe

b. Item 3 should be on a separate line.

c. Write a function that will accept as parameters: a file and a structured record of search values. The function should search all files for the values in the target search record. Such as, if the type is desert, only desert recipes are searched for the other search contents, such as banana in the recipes' lines. Then, display all recipes satisfying the search parameters.

7. **Project code: CC213-07: Maze Search:**

   Write a function that accepts an 8 by 8 array of characters that represents a maze. Each position can contain either an X or a blank. Starting at position (0,1), list any path through the maze to get to location (7,7). Only horizontal and vertical moves are allowed. If no path exists, write a message indicating there is no path. Moves can be made only to locations that contain a blank. If an X is encountered, that path is blocked and another must be chosen. Use recursion.

8. **Project code: CC213-08: Network Addresses Search:**

   Numeric addresses for computers on the international network Internet are composed of four parts, separated by periods, of the form

   xx.yy.zz.mm

   where xx, yy, zz, and mm are positive integers. Locally, computers are usually known by a nickname as well. You are designing a program to process a list of Internet addresses, identifying all pairs of computers from the same locality. Create a structure type called address_t with components for the four integers of an Internet address and a fifth component in which to store an associated nickname of ten characters. Your program should read a list of up to 100 addresses and nicknames terminated by a sentinel address of all zeros and a sentinel nickname.

Sample Data

111.22.3.44	platte
555.66.7.88	wabash
111.22.5.66	green
0.0.0.0	none

The program should display a list of messages identifying each pair of computers from the same locality, that is, each pair of computers with matching values in the first two components of the address. In the messages, the computers should be identified by their nicknames.

Example Message

Machines platte and green are on the same local network.

Follow the messages by a display of the full list of addresses and nicknames. Include in your program a scan_address function, a print_address function, and a local_address function. Function local_address should take two address structures as input parameters and return 1 (for true) if the addresses are on the same local network, and 0 (for false) otherwise.

9. **Project code: CC213-09: Word Lists:**

   Define a structure type to represent a word list. The structure will containone string component for the language of the words (e.g., English, Japanese, Spanish), an integer component that keeps track of how many words are in the list, and an array of MAX_WORDS 20-character strings to hold the words. Define the following functions to work with word lists:

a) load_word_list—Takes as parameters the name of an input file and a wordlist structure to be filled.

b) add_word—Takes as parameters a word and a wordlist structure to modify. If the word list is already full, it displays the message "List full, *word* not added." If the word is already in the list, it leaves the structure unchanged. Otherwise, it adds the word to the list and updates the list size. Do not bother keeping the list in order.

c) contains—Takes as parameters a word and a wordlist. If the word matches one of the wordlist entries, the function returns true, otherwise false.

d) equal_lists—Takes two wordlists as parameters and returns true if the lists are in the same language, have the same number of elements, and every element of one list is found in the other. (*Hint*: Call contains repeatedly.)

e) display_word_list—Displays all the words of its wordlist structure parameter in four columns.

Write a program that fills a wordlist from a data file. Then prompt the user to enter a language and 12 words to add to a different list. Then ask the user to enter some words to search for in the first list using contains, and print a message indicating whether each is found. Use equal_lists to compare the two lists, printing an appropriate message. Finally, use display_word_list to output each list.

10. **Project code: CC213-10: Aircraft Database:**
    Develop a database inquiry program to search a binary file of aircraft data sorted in descending order by maximum cruise speed. Each aircraft record

should include the name (up to 25 characters), maximum cruise speed (in km/h), wing- span and length (in m), the character M (for military) or C (for civilian), and a descriptive phrase (up to 80 characters). Your system should implement an interface that allows the user to search on all components except the descriptive phrase. Here are three planes to start your database:

```
SR-71 Blackbird                       (name)
3500                                  (max cruise speed)
16.95 32.74 M                         (wingspan,length,military/civilian)
high-speed strategic reconnaissance
EF-111A Raven
2280
19.21 23.16 M
electronic warfare
Concorde
2140
25.61 62.2 C
supersonic airliner
```