PROGRAMMING PROJECT#5: INFORMATION AND SUBMISSION FORM - - The "GAME OF LIFE" program (60pts) - due NO LATER than Friday May 15, 2015

This assignment information sheet is to be used ONLY by Santa Rosa Junior College students currently enrolled in CS 10 Sections 4293, 4524 and 5393 (Spring 2015).

Prior to working on this programming project....please read Gaddis textbook Chapter#8 to understand the set-up, structure and typical types of processing done with **User-Defined Multi-Dimensional arrays** and review-reference section 8.9 pgs 545 - 552. You may skip your reading of section 8.11 - "Introduction to STL vector" during your review of the content provided in this chapter. STL vectors are a more advanced topic and beyond the scope of the current CS10 Title V course outline.

It is recommended that you continue with your reading of the weekly lesson note on **Two Dimensional Arrays** on the class schedule pages and review any chapter resources provided on the **class calendar** for weeks#16 - 17. Check the **class downloads and resources page** for any additional ancillary material or project related "hints" provided by your instructor on the current topic under discussion for the week. The project can be a bit challenging, so allocate enough time to (a) set-up two-dimensional arrays, (b) write the algorithms as functions used to implement the criteria on this project and finally time needed to (c) test, debug your project to ensure accurate grid data output for the generations.

Originality of Work : Projects and assignments that show evidence of lack of originality of work in any of the following forms will receive a grade of 0.
Collaboration: Although discussing assignments and projects with fellow students is encouraged, these discussions should not be about specific code and one student should never look at another student's actual code. If significant portions of your code are similar to another student's code, both parties will receive a grade of 0 on the assignment. This is not an hypothetical situation. It happens every semester. If you find yourself hopelessly stuck, get help from me. In summary, there are three things that you should never do:

1. look at another student's code,
2. let another student look at your code, or
3. discuss code with another student in enough detail that parts of your program look identical.

Previous Semester Coursework: You should also never look at a solution to a programming exercise or a project from a previous semester, whether it is a sample solution or a student submission.
Other Sources: Many of the programs that you do this semester are classic problems that have been solved by thousands of students, and some may be found in various textbooks. If you find a book (or website, or other source) that includes a solution or partial solution to one of our assignment, you may review the source code but do not simply copy/paste the code into your program. Instead recreate your own algorithm and source code prior to submitting the assignment or project.

Instructions to all sections: Use your compiler to write a single C++ program following the program specifications detailed below. After successfully compiling and debugging (if needed) your program, paste your source code and contents of your program output in the text input areas provided for the programming segments of this assignment. Complete all parts of this programming assignment and check for errors, prior to clicking submit.

Please make sure you have correctly input your name, section#, email address, SID (last four digits of your srjc student identification number) and the correct web form code visible to you. Upon submission you should receive a confirmation message. If you need further clarification please DO NOT hesitate to contact me, click to **email me** .

IMPORTANT NOTE*: LATE or resubmission of Project#5 passed the due date specified on the submission form will NOT be accepted. NO exceptions!

STUDENT NAME: _____ **CS10 SECTION#:** _____

EMAIL ADDRESS: _____ **SID#:** _____ (last four digits)

Project#5 - Two-dimensional array operations: Game of Life program**

The game of life is a computer simulation of the life and death events of a population of organisms. This program will determine the life, death, and survival of bacteria from one generation to the next, assuming the starting grid of bacteria is generation zero (0). Each cell has a total of up to 8 neighbors, including the 4 immediately adjacent cells and the 4 diagonal cells. The rules for the creation of each cell in the next generation are as follows:

1. If the cell is currently empty:
    1. If the cell has exactly three living neighbors, it will come to life in the next generation.
    2. If the cell has any other number of living neighbors, it will remain empty.
2. If the cell is currently living:
    1. If the cell has one or zero living neighbors, it will die of loneliness in the next generation.
    2. If the cell has four or more living neighbors, it will die of overcrowding in the next generation.
    3. If the cell has two or three neighbors, it will remain living.
3. All births and deaths occur simultaneously (make sure you don't get this one wrong!).

First check the following html links that provide context and visual content of data related to this project: **initial grid data**, **grid output including intermediate generations** and **final correct grid output**.

Your task is to write a program that plays the game of life. The size of the grid will be a 20 x 20 square. **Your solution must use a 20 X 20 two-dimensional array. Don't declare a bigger array!** It is permissible, of course, to use a second array of the same size if you find it convenient or helpful to do so.

A "Game_of_life" class (ADT) set-up and implementation is possible but is not required. **DO NOT use pointers or vectors for this project.**

The original grid of bacteria will be supplied to your program from a text file called bacteria.txt. The text file will contain one line of data for each bacteria in the original grid. Each line will consist of a pair of numbers, separated by a space. The first number will indicate the row location of the bacteria and the second number will indicate the column location of the bacteria. Every number in the text file will be between 0 and 19.

After your program has initialized the grid with generation 0, your program must use the criteria provided above and allow life to proceed for 5 generations. Your program should then display the final results for the fifth generation on the screen, using asterisks (*) to represent live bacteria on a grid, along with the following statistical information:

- The number of living cells in row 10.
- The number of living cells in column 10.
- The number of dead cells in row 16.
- The number of dead cells in column 1.
- The number of living cells in the entire board.
- The number of dead cells in the entire board.

*note ...row 10 and column 10 refers to the actual row and column indexes in the 20 x 20 two-dimensional array

Please make sure that you use the following input file **bacteria.txt** and make sure your output matches the **final correct grid output** exactly. You may use the following visual **grid output including intermediate generations** as you code and test your grid data and output. Note: Do not worry if your compiled version of grid output pasted in the text box below appears a bit skewed in the confirmation email. I will be compiling your submitted source code and so should see the correctly formatted grid.

No credit will be awarded for partially done work, incorrect output or algorithmic solution that was acquired elsewhere and does not reflect your programming style, design and

source code implementation.

Paste your program "source code" for Project#5 - Game of Life program in the text input area below:

Paste your program output for Project#5 - Game of Life program in the text input area below:

General comment area: You may use the text area below to provide any comments on this programming project or if you are unclear and have a question on some of the new C++ syntax or semantics related content covered in recent weeks.

Please also include any comments on something new and useful (C++ related) that you discovered while developing and implementing algorithms (source code) to complete this array processing project. *THANK YOU! (Comments are for Instructor use only and are not used for grading purposes)*

Enter web form code NLSQT

reload image

RE-TYPE EMAIL ADDRESS:

Submit Reset

Sujan Sarkar - CS Instructor, Santa Rosa Junior College

Updated Sunday April 19, 2015