1806ICT Individual Assignment

Milestone 2 due end of Week 6 (35 marks)

Milestone 4 due end of Week 11 (65 marks)

This assignment requires you to solve problems and write software independently, and to integrate various aspects of C programming that you have learned this trimester. You must write ALL the C program code. Lecturer and tutors are happy to answer questions on this assignment.

MILESTONE 2: PART 1: (20 marks)

With regard to the PROGRAM DESCRIPTION listed below, produce a program document which describes:

- user requirements;
- derived requirements;
- how data is represented in your program. You will need to describe important data structures used in your program;
- each of the software modules (functions) shown in the form of a structure chart;
- the control interface of the modules.

The document should also include

- the programs input/output (possibly with an operation of the program i.e., a session with the system, including input and output),
- algorithms for each module where needed, and
- test data for each function and for the overall program. Note that testing can include desk checking with data.

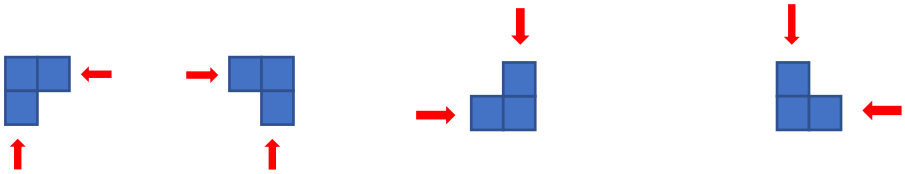
Important:

- You should discuss your ideas with your lecturer or tutor in order to make sure you understand correctly the requirements of the program.
- During the design process, you may find that some requirements are not precise. As a result, you may make some assumptions regarding the operating conditions of your system. You can validate these assumptions with your tutor.

Any assumptions should be stated clearly at the beginning of your program document.

PROGRAM DESCRIPTION

The aim of this question is to produce a program with which a human player can play a simple single player game. The game is played on an n x n board where n is a user input parameter. Initially the whole board is filled with one symbol per cell and the symbols are chosen randomly from a pre-defined list. The task of the player is to swap adjacent symbols such that three or more identical symbols of the same type are grouped together into one of the following four shapes.



Note that only such swaps are allowed that produce one of these four shapes of identical symbols are allowed. A swap that does not achieve this cannot be executed. Swaps can only be in a column or in a row, but not diagonal.

If a swap changes the board such that at three similar symbols are grouped in one of these four shapes then these symbols will immediately disappear. The gaps that these leave behind will be filled by symbols already existing on the board moving in one of the directions shown by the arrows. This of course produces gaps in the board which will be filled with new randomly selected symbols.

Note the user should never see a board that has the same three symbols grouped together in one of the four shapes. It is, of course, possible that a single swap causes three similar symbols to become grouped together in one of the four shapes. In this case these symbols in the group disappear which could cause a "chain reaction" where another of the four shapes appear and need to be dealt with.

Each symbol removed from the board counts 10 points. As the game is played against time, the final score is the quotient of the number points divided by the total playing time in seconds. The user will nominate the number of symbols to be used at the start of the game.

This is a single player game. Your program prompts the user for a move and executes the move (displaying the updated board) if the move is permissible. If the move is not permissible it should be rejected. The user can input a move by typing in the "from" coordinates and the "to" coordinates. Once the move has been executed, your program should display the score obtained from the move as well as the cumulative total score. The player can also choose to start a new game or quit the game at any time with the option of saving the game to disk so that it can be resumed at a later date.

MILESTONE 2: PART 2: (15 marks)

A Word Sum is a puzzle in which the digits in a correct mathematical expression, such as a sum, are replaced by letters and where the puzzle's words are in the dictionary (*dictionary.txt*). For example, if D = 7, E = 5, M = 1, N = 6, O = 0, R = 8, S = 9 and Y = 2 then the following is true:

```
  SEND        =>  9567
  MORE        =>  1085
 ---------        -------
 MONEY        => 10652
```

Two conditions are assumed: firstly, the correspondence between letters and digits is one-to-one and different letters represent different digits. Secondly, the digit zero does not appear as the left-most digit in any of the numbers.

Write a program that, given a Word Sum on the command line, determines the correct assignment of digits to letters so that the numeric summation is correct. So the command to run the program would be: *./a.out send more money*

Demonstrate your program with the following problems:

SEND + MORE = MONEY
EARTH + AIR + FIRE + WATER = NATURE
SATURN + URANUS + NEPTUNE + PLUTO = PLANETS

Required Program Output Format:

Problem: SEND + MORE = MONEY, CPU = 0.158406
D = 7, E = 5, M = 1, N = 6, O = 0, R = 8, S = 9, Y = 2, Attempts = 1110106

Problem: EARTH + AIR + FIRE + WATER = NATURE, CPU = 0.238431
A = 7, E = 6, F = 8, H = 2, I = 0, N = 1, R = 4, T = 3, U = 5, W = 9, Attempts = 1354807

Problem: SATURN + URANUS + NEPTUNE + PLUTO = PLANETS, CPU = 0.161114

A = 2, E = 9, L = 6, N = 3, O = 8, P = 4, R = 0, S = 1, T = 7, U = 5, Attempts = 760286

MILESTONE 4: PART 1: (45 marks)

Code and test the program your program document from Milestone 2 : Part 1 described and demonstrate the workings of your program.

Requirements:

- You are responsible for designing a program which addresses the requirements.
- You should design your algorithms and your overall program so that your code is efficient and clean.
- You should test the components of your program separately, but you should end up with one working program at the end.

MILESTONE 4: PART 2: (20 marks)

Write a C program which reads dictionary.txt and builds a data structure which allows a user to input any 3 letter sequence and quickly get a list of all words containing that sequence. Your program must be efficient with regard to CPU usage and memory usage.

SUBMISSION [VIA L@G]

All filenames must include your surname and your student number

MILESTONE 1: Part 1, Program Document and Part 2, C source file.

MILESTONE 2: Part 1, C source file and Part 2, C source file.