# Diagram Description automatically generatedCairo University

**Faculty of Computers & Artificial Intelligence Structured Programming (2021/2022)**

Assignment 2

**Delivery Notes:**

* This is a group assignment of 2 members (at most) and the members should be from the same group/lab.
* Both students should work and fully understand everything in the code.
* Due date is on Dec 22th until 11:55 pm (try to give yourself at least 1 hour, or more, to handle submission and submission problems).
* No late submission is allowed.
* Submission will be on blackboard.
* No submission through e-mails.
* For each task you will develop a .cpp file that should include a block comment containing students’ IDs and names. These files should be named task1.cpp, task2.cpp, task3.cpp, & task4.cpp then put these 3 files in a folder named GroupNum\_firstStudentID\_SecondStudentID and compress them to a .zip file with the same folder name. The compressed file would be the file to be delivered.
* The allowed values for group numbers in the zip file name is one of the following 8 **groups**: **S1 till S8**. Writing the wrong group will not allow your TA to receive your assignment and you might lose its grade.

# Failing to abide by the naming conventions of the file, would result in a ZERO for both team members.

* **Do not send your code** to anyone**,** so that no other student would take your files and submit it under their names.

# In case of Cheating, you will get a negative grade whether you give the code to someone, take the code from someone/internet, or even send it to someone for any reason. Taking only part of the code from a friend is also cheating.

* You have to write clean code and follow a good coding style including choosing meaningful variable names.
* **You have to remember every part of the code and fully understand it during the discussion. Forgetting any part of the code for any reason won’t be excused.**

**Important notes:**

* You cannot use external libraries to complete your code, except for rand(), srand(), time() for random number generation.
* Write each part of the code that does a certain objective in a function. As there will be grades deducted for not using functions appropriately. (Do not attempt to outsmart this rule, for example writing one function above the main and calling it to do all the work is not a correct way of using functions).  
  An appropriate use would be to group each few lines of code that does a certain functionality into a function. Most of the required functions will be mentioned below.

# Problem-1: Draw a rectangle:

Implement a function that draws a rectangle shape based on the given dimension. This function will take two parameters:

* L: Length of the rectangle
* W: Width of the rectangle

Conditions:

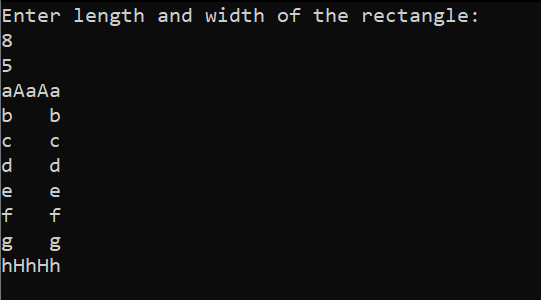
* Output shape should be printed using the alphabets, starting with the alphabet ‘a’ on the first line, then ‘b’ on the second, ‘c’ on 3rd and so on.
* The casing of the alphabet should change in each column (**only columns**). The first column will have lower case, the 2nd upper case, 3rd lower case again and so on.
* The inside of the rectangle should be empty (fill it with spaces).

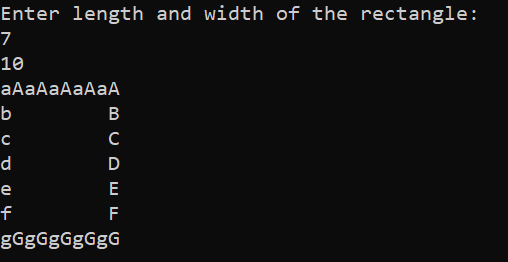
The function that draws the rectangle will call 2 functions to help it perform its task.

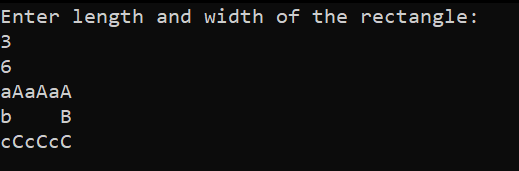
The first function will be called to handle drawing the top or bottom line. The second function will be called to draw middle lines.

(Hint: the lower and upper case ascii of an alphabet character are 32 characters apart).

Examples:







# Problem-2: Serial code:

[You may use arrays, though they aren’t necessary]

Clarification: alphabets/letters are (a-z, A-Z), digits (0-9). Characters in this problem are both alphabets and digits.

The serial code used to verify the product key for a software is 20 character, every 5 character separated by a dash ‘-’ (AAAAA-BBBBB-CCCCC-DDDDD). We’ll call every 5 characters a segment (for example the BBBBB part is a segment, the DDDDD is a segment). The serial is made up of 4 segments (20 characters), the dashes do not count toward the number of characters of the serial and is only visual. The serial is made up of numbers and characters. The user can ask you to (write them in a function and call them in the main):

1. Generate a new random serial code (the verification rules apply to generating the serial code). The number has to be randomly generated (can’t be constant, or from a list). The serial will be printed with the dashes included.
2. Enter a serial code to verify (the serial will be entered by the user with the dashes included).

To verify a serial code, you must be sure it follows the following rules (In addition to it being 20 characters, 4 segments and write each validation rule in a function and call them in the code whenever needed):

1. The total number of digits found in all the first characters of each segment should be even.
2. The number of alphabets in a single segment should be at least 1.
3. The sum of the digit values in all the serial should be more than 40
4. The sum of digits in the last segment (4th) should be less than 15.
5. The 3rd and 12th characters should be digits, they should also both add up to 9.
6. There should be at least 3 capitalized letters, and 2 lower case letters in all the serial.
7. Every alphabet character (except the first one encountered) should be at least 4 steps away from the previous one, going forward or backward in the alphabet. (‘z’, ‘a’ are considered 1 character away from each other, consider the characters connected at the start and end, so after abcde…xyzabcd… like a being drawn on circle/clock).   
   For example: 62Y4B-47… is wrong, because ‘B’ is 3 steps away from ‘Y’ when going backwards (B to Y), even though B to Y going forward is 23 steps, but the distance has to be from both sides.  
   54X03-D9rA… is correct, because D is far enough from X going forward D to X or backward X to D, and ‘r’ is far enough from ‘D’, A is far enough from R.  
   Note that the letter ‘r’ being lower case still counted as an alphabet. And changing from lower to upper or upper to lower does not change anything in the rule, you still keep the distance regardless of the casing

Please not you have to generate codes too that follow these rules.

For validation, write every one of the rules (1 to 7) as a function that you can call to validate the rule on the given code.

Example code that follows the rules:  
a9898-1G343-m1Y3R -77Eqc

**Rule 1**: even number of digits counting the first character in each segment (a, 1, m, 7 🡪 2/4 are digits, 2 is an even number). [a9898-1G343-m1Y3R -77Eqc]  
**Rule 2**: every segment has at least1 alphabet.[ **a**9898-1G343-m1Y3R -77Eqc]  
**Rule 3**: sum of digits is (9+8+9+8+1+3+4+3+1+3+7+7) = 63 which is more than 40. [a9898-1G343-m1Y3R -77Eqc]  
**Rule 4**: 7+7 = 14 which is less than 15. [a9898-1G343-m1Y3R -77Eqc]  
**Rule 5**: 3rd, 12th character are digits and add up to 9 (8+1). [a9898-1G343-m1Y3R -77Eqc]  
**Rule 6**: at least 3 caps and 2 lower case. [a9898-1G343-m1Y3R -77Eqc]  
**Rule 7**: distance of 4 or more steps in the alphabet between consecutive characters: (a,g)=6, (g,m)= 6, (m,y) = 12, and so on, all of them being at least 4 characters away. [a9898-1G343-m1Y3R -77Eqc]