CO222: Programming Methodology

Project 1: QR Code Generator - Specification

Objectives

Making you familiar with,

- Arrays
- Loops
- Functions
- Command-line argument handling
- Debugging
- Good coding practices
- Linux

Introduction

Given a URL you have to encode it to a QR (Quick Response) Code (According to the given algorithm) and print it on the Linux terminal. You will be given a sample program that would do the same task. Your task is to replicate the functionality of the given program.

The output should print exactly at the same place and scale as the given sample program.

Please note that all the algorithms which are given are made up algorithms, you cannot reverse the QR to the string by the existing QR scanners.

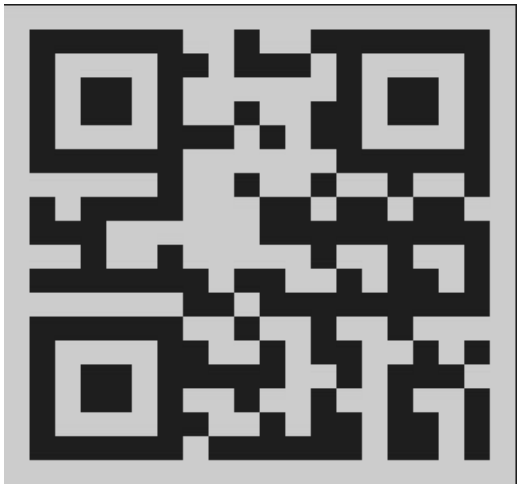
Inputs:

User inputs:

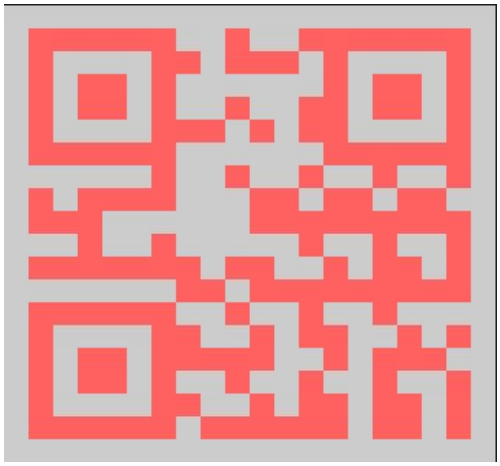
- You should get the URL as a user input through STDIN (this string should be greater than 3 characters and less than 120 characters)

Command-line arguments:

- If a colour is given as an argument to the program with the '-c' flag, the program should be able to print the QR code using the given colour. If no argument is given use black and white as default.



Eg: ./program -c red
Given the above arguments, the program should print the QR code in red and white.

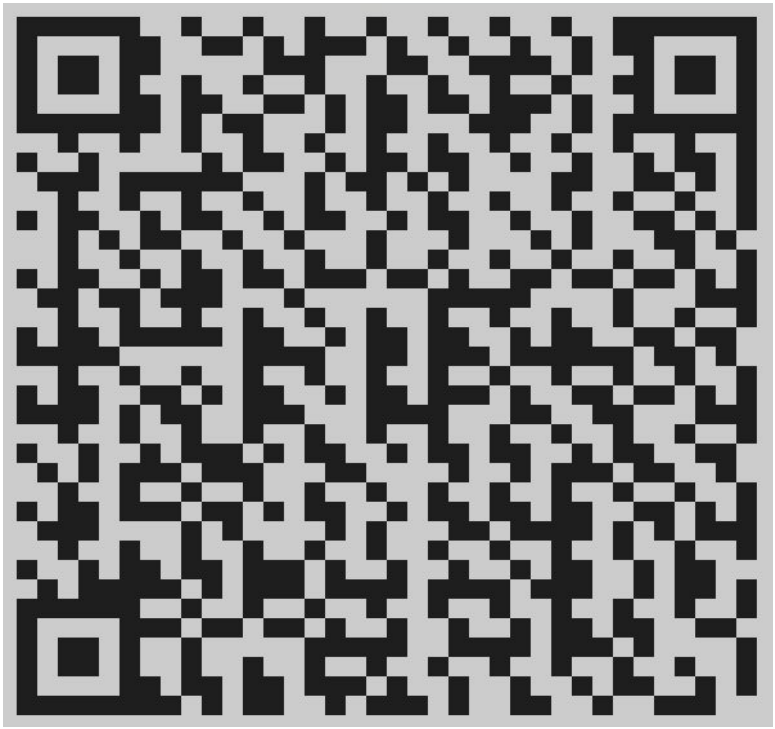


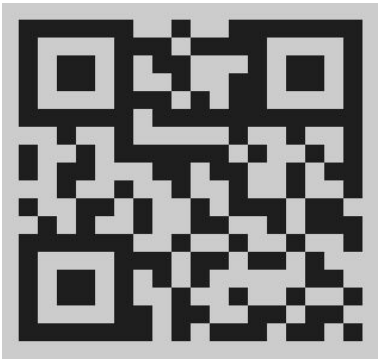
These colors should be supported by the program - black, red, green, yellow, blue, magenta, cyan

- If -h flag is given or the given arguments are invalid, you should print the usages of your program (how the arguments should be given). See the sample binary given for the expected behaviour.

Output:

Print the relevant QR code in the terminal. Consider the length of the string and choose to which version the string should be encoded. If the input string is less than or equal 20 characters it should be encoded to the smaller version as in the 2nd figure and if the number of characters of the input string is more than 20 it should be encoded to the larger version as in 1st figure.





String Hashing algorithm

To encode a string to QR code, first, you need to hash the string and convert it to a fixed-length string. For the smaller QR version (Input string length less than or equal 20) you need to hash the string to 24 characters and for the larger version(Input string greater than 20) you need to hash the string to 132 characters. Follow the below steps to get the hashed string.

The ASCII value of the first character of the hashed string = Length of the input string + 50 (which is used for decoding purposes).

Assume the length of the input string is L. Then, the next L characters of the hashed string should be the characters of the input string. To fill the remaining characters (this part of the string is used for error detection), add 1 to the ASCII values of characters in the input string and append them to the string until you reach the required length. Do this by incrementing the number you are adding by 1 in each round until all the required number of characters is obtained. Then reverse the error detection part of the hashed string.

Observe the given example:

Input string: ABCDEFGH
First character = 50 + 8

58	65	66	67	68	69	70	71	72	66	67	68	69	70	71	72	73	67	68	69	70	71	72	73	ASCII
	A	B	C	D	E	F	G	H	A +1	B +1	C +1	D +1	E +1	F +1	G +1	H +1	A +2	B +2	C +2	D +2	E +2	F +2	G +2	

Reverse this part of the array

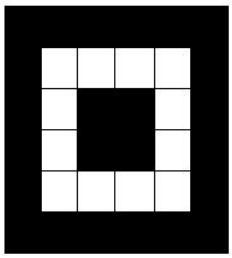
58	65	66	67	68	69	70	71	72	66	67	68	69	70	71	72	73	67	68	69	70	71	72	73

58	65	66	67	68	69	70	71	72	73	72	71	70	69	68	67	73	72	71	70	69	68	67	66

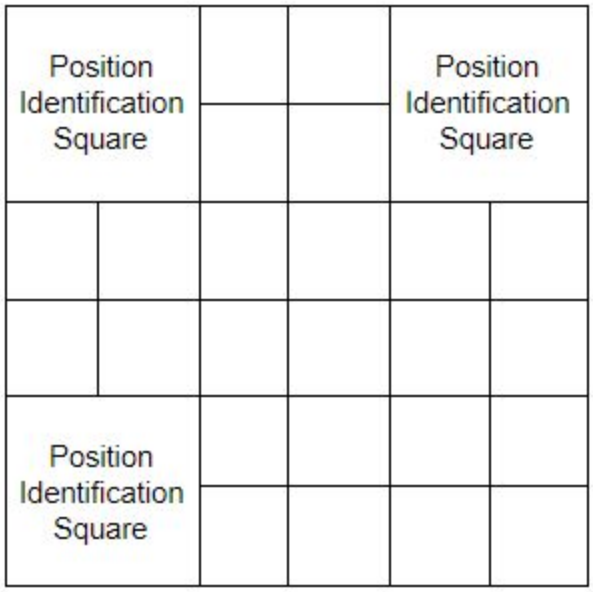
58	65	66	67	68	69	70	71	72	73	72	71	70	69	68	67	73	72	71	70	69	68	67	66	ASCII
:	A	B	C	D	E	F	G	H	I	H	G	F	E	D	C	I	H	G	F	E	D	C	B	Hashed string

QR generation algorithm

Each QR Code has 3 Position identification squares (6x6 unit squares) in 3 corners. They should be drawn in the pattern given below:



The remaining area is for character encoding, which is divided into small squares. (of size 3x3 unit squares each).



Each small square represents one character of the hashed string. Figures below represent which character should be encoded within each square in the 2 different versions.

The QR has a white border(of width 1 unit square) for clarity, observe the outputs of the sample program and implement the exact same thing. Considering scale position and color.

- The small version - for string length 20 or less (First get the hashed string of length 24)

Position		ch17	ch18	Position	
		ch19	ch20		
ch21	ch22	ch1	ch2	ch3	ch4
ch23	ch24	ch5	ch6	ch7	ch8
Position		ch9	ch10	ch11	ch12
		ch13	ch14	ch15	ch16

- The larger version - for string length above 20 (first obtain the hashed string of length 132)

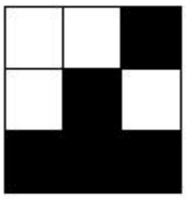
Position		c101	c102	c103	c104	c105	c106	c107	c108	Position	
		c109	c110	c111	c112	c113	c114	c115	c116		
c117	c118	ch1	ch2	ch3	ch4	ch5	ch6	ch7	ch8	ch9	ch10
c119	c120	ch11	ch12	ch13	ch14	ch15	ch16	ch17	ch18	ch19	ch20
c121	c122	ch21	ch22	ch23	ch24	ch2	ch2	ch2	ch2	ch2	ch3
c123	c124	ch3	ch3	ch3	ch3	ch3	ch3	ch3	ch3	ch3	ch4
c125	c126	ch4	ch4	ch4	ch4	ch4	ch4	ch4	ch4	ch4	ch5
c127	c128	ch5	ch5	ch5	ch5	ch5	ch5	ch5	ch5	ch5	ch6
c129	c130	ch6	ch6	ch6	ch6	ch6	ch6	ch6	ch6	ch6	ch7
c131	c132	ch7	ch7	ch7	ch7	ch7	ch7	ch7	ch7	ch7	ch8
Position		ch8	ch8	ch8	ch8	ch8	ch8	ch8	ch8	ch8	ch9
		ch9	ch9	ch9	ch9	ch9	ch9	ch9	ch9	ch9	c100

After finding the relevant character of the square, to encode it you should get the ASCII value of the character and then get the binary representation using 9 bits. Then colour each unit square of the 3x3 square according to the value of the corresponding bit. Black if the value is 1 and white otherwise. (Consider MSB as b1 and LSB as b9)

b1	b2	b3
b4	b5	b6
b7	b8	b9

Eg 'W' ASCII value = 87 Binary representation: 001010111

The encoded pattern for W:



Problem breakdown

Instead of directly going to code the solution, you must break down the problem into smaller subproblems, such as

1. How to get user input
2. How to hash the string to a fixed-length
3. How to print basic frame and position identification squares
4. How to print the encoded pattern of one character
5. How to print multiple patterns in different positions of the console.
6. How to assign a color when printing
7. Error handling and the argument handling

Instead of thinking about the problem as a whole, try and find the solutions for each one of the above questions. That will help you to reach the final goal smoothly and accumulatively.

Important

- Pay attention to boundary cases and handle the errors correctly.
- Refer ANSI escape codes which will be useful to complete this project.
- Under no circumstance, you should copy somebody else's code. Copying someone else's code (including your group mate's) or showing your source code to anyone else will earn you zero mark for the whole project. You might need to be really careful because this has happened many times in the past. Therefore, put some honest effort to earn the marks for project 1.
- Basic functionality, functionality with colors, error handling, modularization (functions) and good coding practices will be considered when marking.

Deadline

The deadline for the submission is **24th April 2020** 23:55h (No late submissions are accepted!)
Submit your answer (without compile error or warning) in a single file with the filename E17yyyproject1.c, where yyy is your registration number.
Marks: You will get 10 marks for your final grade from the project.