

CS101 Homework 3

# Data Matrix Generation

Oct. 31, 2022

**Due** Nov. 7, 2022 (Mon)

If you have any questions, use the Homework 3 Q&A board at Elice.

## 1 Overview

A Data matrix is a kind of 2D barcode, which is a visual, machine-readable data representation that can contain various information. Like other 2D barcodes, such as QR codes, Data matrix codes are widely used in product identification. Especially, they are used to mark small items such as small instruments, medicine, and electronic devices because of their small size. When they contain the same amount of information, a Data matrix is much smaller in physical size than a QR code. For example, the Data matrix is much smaller than the QR code in Fig. 1b even though both codes contain identical data and their unit cell size is the same.

In this homework, you will generate a  $10 \times 10$  Data matrix that can contain three alphanumeric characters. You can read the generated Data matrix using a Data matrix reader application your smartphone. In Task 1, you should convert an input string to binary data, because the Data matrix can only represent binary values in each cell (black or white). In Task 2, you should draw the Data matrix image that encodes the binary data. In Task 3, you should read and decode a Data matrix image.



Figure 1. Examples of Data matrix codes

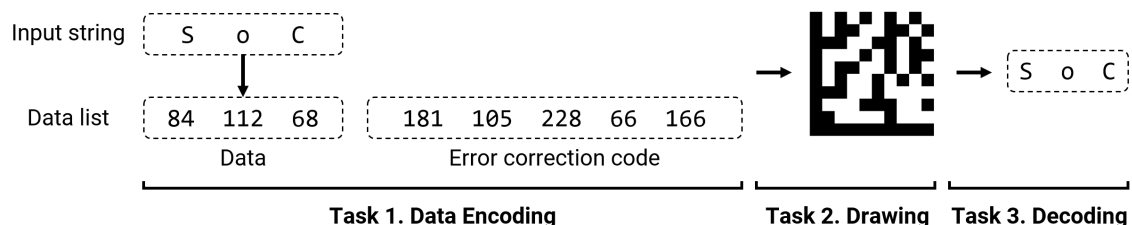


Figure 2. Homework 3 Overview

## 2 Preliminaries

### 2.1 Image Zoom

You will use the `cs1media` library used in Lab 4. Because the size of Data matrix is small, you can zoom the image by clicking Zoom menu.

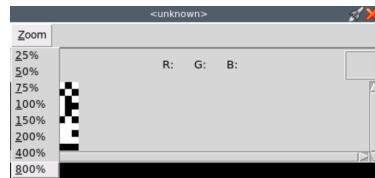


Figure 3. `cs1media` Image Window

### 2.2 ASCII Codes

Since a computer is an electronic device, it can only store binary data represented by the flow of current. Therefore, every data in a computer is saved as binary. While converting integer data into binary is trivial, this does not apply in the same way for string data. In this case, we use the ASCII code. In, ASCII code, a single number corresponds to a single alphabet.

#### ASCII Table

Dec = Decimal value, Chr = Character / 0~31: Special characters

Dec	Chr	Dec	Chr	Dec	Chr	Dec	Chr	Dec	Chr	Dec	Chr	Dec	Chr
0	NUL (null)	16	DLE (data link escape)	32	SPACE	48	@	64	P	80	p	96	~
1	SOH (start of heading)	17	DC1 (device control 1)	33	!	49	A	65	Q	81	q	97	a
2	STX (start of text)	18	DC2 (device control 2)	34	"	50	B	66	R	82	r	98	b
3	ETX (end of text)	19	DC3 (device control 3)	35	#	51	C	67	S	83	s	99	c
4	EOT (end of transmission)	20	DC4 (device control 4)	36	\$	52	D	68	T	84	t	100	d
5	ENQ (enquiry)	21	NAK (negative acknowledge)	37	%	53	E	69	U	85	u	101	e
6	ACK (acknowledge)	22	SYN (synchronous idle)	38	&	54	F	70	V	86	v	102	f
7	BEL (bell)	23	ETB (end of trans. block)	39	'	55	G	71	W	87	w	103	g
8	BS (backspace)	24	CAN (cancel)	40	(	56	H	72	X	88	x	104	h
9	TAB (horizontal tab)	25	EM (end of medium)	41	)	57	I	73	Y	89	y	105	i
10	LF (NL line feed, new line)	26	SUB (substitute)	42	*	58	:	74	J	90	z	106	j
11	VT (vertical tab)	27	ESC (escape)	43	+	59	;	75	K	91	{	107	k
12	FF (NP form feed, new page)	28	FS (file separator)	44	,	60	<	76	L	92		108	l
13	CR (carriage return)	29	GS (group separator)	45	-	61	=	77	M	93	}	109	m
14	SO (shift out)	30	RS (record separator)	46	.	62	>	78	N	94	~	110	n
15	SI (shift in)	31	US (unit separator)	47	/	63	?	79	O	95	_	111	o

Figure 4. ASCII Table

In Python, you can convert a string to integer using the `ord()` function, and an integer to string using the `chr()` function. For example, `ord('A')` returns 65, and `chr(65)` returns 'A'.

In addition, you can convert a decimal integer to binary using the `bin()` function. The function returns a string representing the binary value with the prefix '0b'. For example, `bin(17)` returns '0b10001'.

### 2.3 Error Correction Code

Since 2D barcode is a camera-based image, errors can occur if some part of the barcode is shaded, contaminated, or distorted. Therefore, most 2D barcode contains error correction code (ECC). ECC allows the barcode to be decoded correctly although some parts of the barcode are recognized wrong.

Data matrix uses Reed-Solomon ECC for error correction. Since generating ECC is challenging, you can use the provided `ecc()` function in `cs101_hw3` module to get ECC. This takes an integer list of length 3 as input, and outputs ECC list of length 5. For example, `ecc([84, 112, 68])` returns [181, 105, 228, 66, 166].

## Task 1. Data Encoding (10 pts)

A Data matrix consists of black and white cells which represent bits, the most basic unit of data. In other words, each cell represents a binary value. A black cell represents a 1 and a white cell represents a 0. Therefore, you first need to convert an input string into binary data. In this task, you convert an input string to a list of integers, append a Reed-Solomon ECC to the list, and finally convert it to a binary data list.

### Task 1.1 (3 pts)

Implement the `str_to_list()` function that converts a given string to a list of integers. Each element of the list should be the addition of 1 and the ASCII code number of the corresponding character in the string, i.e.,  $1 + (\text{ASCII code number})$ . If the length of string is bigger than three, convert only the first three characters. If the length of string is less than three, pad with spaces (' ') at the end of the string to make its length be three. (e.g., 'H' to 'H ')

**Input** An alphanumeric string

**Output** A list of three integers

**Example**

Input	Output
SoC	[84, 112, 68]
CS101	[68, 84, 50]
H	[73, 33, 33]

### Task 1.2 (2 pts)

Implement `append_ecc()` function that appends a Reed-Solomon ECC at the end of the integer list you created in Task 1.1.

**Input** A list of three integers

**Output** A list of eight integers with a Reed-Solomon ECC

**Example**

Input	Output
[84, 112, 68]	[84, 112, 68, 181, 105, 228, 66, 166]
[68, 84, 50]	[68, 84, 50, 194, 125, 105, 254, 200]
[73, 33, 33]	[73, 33, 33, 76, 146, 191, 100, 162]

### Task 1.3 (5 pts)

Implement `list_to_binary(s)` that converts the integer list with ECC you created in Task 1.2 to the binary data list, a list of eight 8-bit binary tuples. Each element of the list should be a tuple with 8 integers, which represents the binary value of corresponding character. For example, 84 ('S') should be converted into (0, 1, 0, 1, 0, 1, 0, 0).

**Input** A list of eight integers with a Reed-Solomon ECC

**Output** A binary data list, a list of eight 8-bit binary tuples

**Example**

Input	Output
[84, 112, 68, 181, 105, 228, 66, 166]	[(0, 1, 0, 1, 0, 1, 0, 0), (0, 1, 1, 1, 0, 0, 0, 0), (0, 1, 0, 0, 0, 1, 0, 0), (1, 0, 1, 1, 0, 1, 0, 1), (0, 1, 1, 0, 1, 0, 0, 1), (1, 1, 1, 0, 0, 1, 0, 0), (0, 1, 0, 0, 0, 0, 1, 0), (1, 0, 1, 0, 0, 1, 1, 0)]

## Task 2. Drawing Data Matrix (20 pts)

A Data matrix consists of two parts: a finder pattern and a data region. (See Fig. 5(a)) The finder pattern consists of an alignment pattern and a clock pattern. The alignment pattern is the L-shaped black line at the left and the bottom of the Data matrix, and the clock pattern is the line with alternating white and black cells at the top and the right of the Data matrix. The alternating pattern should start with a white cell from the cell on the upper right corner. The data region is the region inside the finder pattern, which encodes the data. When reading a Data matrix, the camera first detects the Data matrix by finding the finder pattern and then reads the data region to decode the data.

The data region consists of data blocks which consist of 8 cells each. Specifically, a  $10 \times 10$  Data matrix has 8 data blocks. In each block, the binary data is filled from the upper left cell to the right. If a row is full with data, the next row is filled from left to right. For example, 01010100 is filled as Fig. 5(b).

In a  $10 \times 10$  Data matrix, eight data blocks are placed in the manner shown in Fig. 6. If a block exceeds the data region while placing, it should be placed at the opposite side of the data region.

In this task, the position of a block is defined as the top-left position of the block. For example, the first block is placed at (7, 3). Beware that the coordinate starts from (0, 0), the top-left corner of the data matrix including the finder pattern.

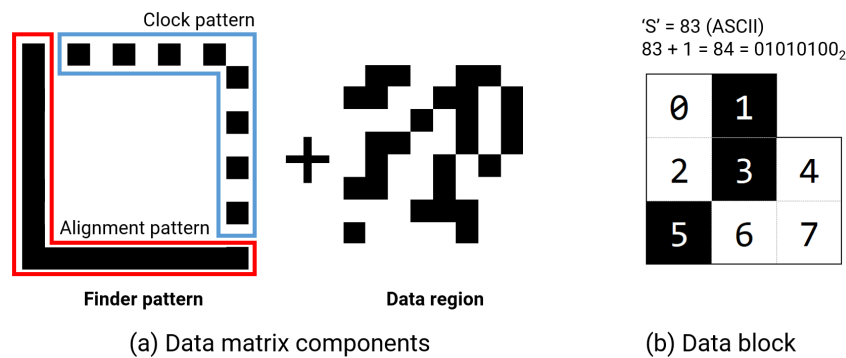


Figure 5. Structure of Data matrix and data block. In (b), the numbers on cells denotes the order.

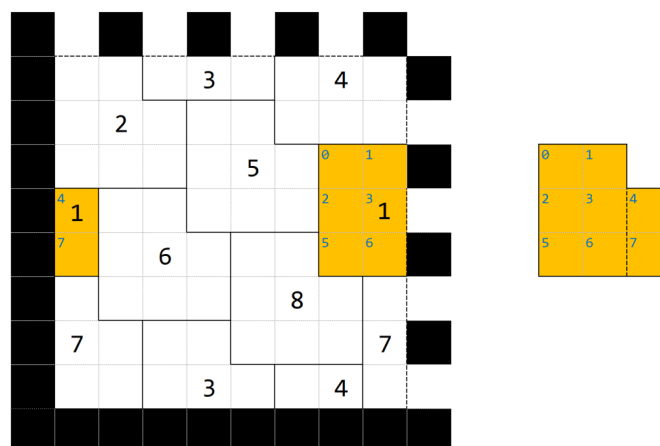


Figure 6. Data block placement. Black numbers denote the placement order, and small blue numbers denote the cell order in Fig. 5.

Task 2.1 (10 pts)

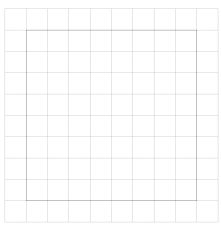
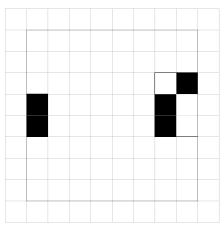
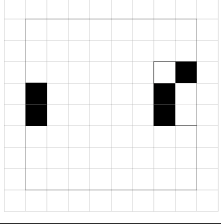
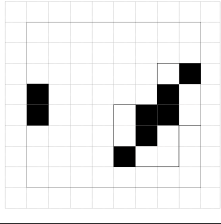
Implement the `place_block()` function that places a data block represented in a 8-bit binary tuple at the specified position `(x, y)` on the  $10 \times 10$  empty image.

**Input**

- `img`: A  $10 \times 10$  `cs1media` image
- `binary_data`: A 8-bit binary tuple
- `position`: A coordinate tuple `(x, y)`

**Output** A  $10 \times 10$  image with the data block placed

**Example**

Input			Output
img	binary_data	position	
	(0, 1, 1, 0, 1, 1, 0, 1)	(7, 3)	
	(0, 1, 0, 1, 0, 1, 0, 0)	(5, 5)	

\* The grid in the example image will not be included in the input.

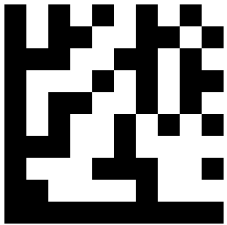
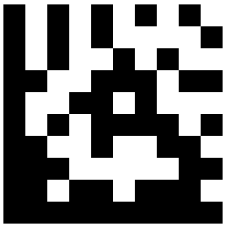
Task 2.2 (10 pts)

Implement `create_data_matrix()` that converts the given string into a  $10 \times 10$  Data matrix. If the length of the string is not three, preprocess it as in Task 1.1. You can copy-and-paste the code you implemented in Task 1 to use it for this task. (10 pts)

**Input** An alphanumeric string

**Output** A  $10 \times 10$  Data matrix image

**Example**

Input	Output	Input	Output
SoC		CS101	

### Task 3. Decoding (20 pts)

Finally, we start from the Data matrix and decode it into the string.

#### Task 3.1 (10 pts)

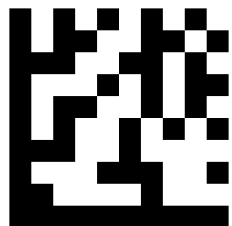
Implement `read_data_block()` that reads a data block at position  $(x, y)$ .

**Input**

- `data_matrix`: A  $10 \times 10$  Data matrix image
- `position`: A coordinate tuple  $(x, y)$

**Output** A 8-bit binary tuple

**Example**

Input		Output
img	position	
	(7, 3)	(0, 1, 0, 1, 0, 1, 0, 0)

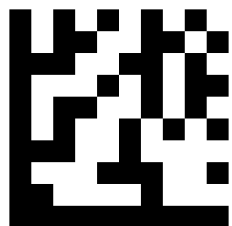
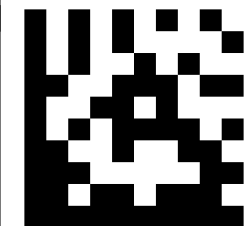
#### Task 3.2 (10 pts)

Implement `read_data_matrix()` that decodes a given data matrix into string. You don't need to check the ECC, that is, you only have to decode three blocks. Do not remove the padding at the end of the decoded string.

**Input** A  $10 \times 10$  Data matrix image

**Output** A decoded string of length 3

**Example**

Input	Output	Input	Output
	SoC		CS1

If you are interested in the Data matrix format, refer to the following materials.

- ISO/IEC 16022 standard
- [https://www.keyence.co.kr/ss/products/auto\\_id/barcode\\_lecture/basic\\_2d/datamatrix/](https://www.keyence.co.kr/ss/products/auto_id/barcode_lecture/basic_2d/datamatrix/)
- [https://en.wikipedia.org/wiki/Data\\_Matrix](https://en.wikipedia.org/wiki/Data_Matrix)