Middle East Technical University
Department of Computer Engineering

CENG 229

Fall 2018

Preliminary for Lab Exam 2

REGULATIONS

Submission and Attendance: Electronically. You will be submitting (or evaluating) your program source code written in a file named `lab2.c` through the CENGServis web system in the lab. You need to click evaluate button (or submit a file) at least one time to get the attendance points. Resubmission is allowed, the last will replace the previous.

Team: **This is an exam.** There is **no** teaming up. The lab exam has to be done/turned in individually.

Cheating: We have zero tolerance policy for cheating. All parts involved (source(s) and receiver(s)) get zero. You will face disciplinary charges.

PROBLEM

There are major differences between how we see an image and how a computer sees it. From the perspective of the computer, an image consists of tiny elements called pixels. A float value that varies from 0 to 255 indicates the color of the pixel at a particular location on a grid. An example image grid is given below with its indices. Horizontal axis (x-coordinate) indices grow from left to right and vertical axis (y-coordinate) indices grow from bottom to top.

7	150	16	18	195	255	25	84	79
6	146	43	170	190	46	32	97	88
5	135	90	43	29	55	23	203	110
4	100	110	31	8	13	10	200	129
3	80	102	78	19	10	170	171	183
2	176	170	90	164	140	150	170	168
1	56	128	232	90	110	68	150	165
0	47	213	60	80	100	120	135	150
	0	1	2	3	4	5	6	7

You are expected to implement a function using the image notation specified above. In this function, you will translate a subsection of an image to another region.

SPECIFICATIONS

- You are expected to implement the `translate` function. This function will take the pixel information from the area to be translated and will write it over the new region.
- The area to be translated and the region to be written over are denoted by their lower left corner (x and y coordinates), width and height. Width and height are the numbers of pixels the area covers along the x-axis and y-axis, respectively.
- The function will take the 2D array representation of **a square** image, **one of its dimensions**, x and y coordinates for lower left point, width and height of the area to be translated and lastly x and y coordinates of the lower left corner of the new region as its arguments. See Table 1.
- You need to relocate the pixel information. While doing so, you will make changes on the given image array.
- **You should solve this problem in place. That is to say, it is not allowed to use a separate memory, an array different than the provided one or dynamically allocated memory, in your solutions.**

56	57	58	59	60	61	62	63
48	49	50	51	52	53	54	55
40	**41**	**42**	**43**	**44**	45	46	47
32	**33**	**34**	**35**	**36**	37	38	39
24	**25**	**26**	**27**	**28**	29	30	31
16	**17**	**18**	**19**	**20**	21	22	23
8	**9**	**10**	**11**	**12**	13	14	15
0	1	2	3	4	5	6	7

56	57	58	59	60	61	62	63
48	49	50	**41**	**42**	**43**	**44**	55
40	41	42	**33**	**34**	**35**	**36**	47
32	33	34	**25**	**26**	**27**	**28**	39
24	25	26	**17**	**18**	**19**	**20**	31
16	17	18	**9**	**10**	**11**	**12**	23
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7

Table 1: Example translation from **point** (1,1) to **point** (3,2)

- Let A be the original region and A' be the translated region. It is given that **the cardinality of** the difference between the whole region R and the union of A and A' will be greater than or equal to **the cardinality of** the intersection of A and A'.

$$|R - (A \cup A')| \geq |A \cap A'|$$

- **You can not make any changes on the pixels positioned at the region $R - A'$. In other words, the final values of those pixels must be the same as their initial values.**
- You are provided a source code with a number of test cases which you should take into consideration. In this source code, an image is represented as a 2D integer array and filled with distinct integers. So that it can be easily debugged. In addition, `print_image` function is provided and implemented for your convenience. `print_image` function prints the pixel values in such an order that the value of the pixel $image[0][0]$ is placed at the bottom left corner.
- Do not attempt to change the signature of the function. The function will be graded by an automatic grader. Therefore you should absolutely comply with the specifications. Please check the Example Runs section.

- **Tips:** You can use array indexing while accessing to image pixels. You can change the size in the source code to generate images in different sizes.

EVALUATION

- There will be no erroneous test input. For example, there will be no out of bounds case for both original and translated region.
- **There will be no degenerate (a rectangle where one dimension is zero) rectangles. (Neither for R nor for A)**
- All submissions will be compiled and run under strictly equal conditions on multiple data sets.
- Grading is based on automatic evaluation (blackbox testing by a software) with a similar yet different input than the one given during the lab; and in certain cases, we may review your code manually to see if it obeys all the requirements of the lab assignment.

EXAMPLE RUNS

Following examples use the initial image array given below

56	57	58	59	60	61	62	63
48	49	50	51	52	53	54	55
40	41	42	43	44	45	46	47
32	33	34	35	36	37	38	39
24	25	26	27	28	29	30	31
16	17	18	19	20	21	22	23
8	9	10	11	12	13	14	15
0	1	2	3	4	5	6	7

Example 1
Function Call:
translate(img, 8, 4, 4, 2, 2, 2, 0);
Result of print_image:

```
056 057 058 059 060 061 062 063
048 049 050 051 052 053 054 055
040 041 042 043 044 045 046 047
032 033 034 035 036 037 038 039
024 025 026 027 028 029 030 031
016 017 018 019 020 021 022 023
008 009 044 045 012 013 014 015
000 001 036 037 004 005 006 007
```

Example 2
Function Call:

```
translate(img, 8, 2, 1, 3, 5, 3, 1);
```

Result of print_image:

```
056 057 058 059 060 061 062 063
048 049 050 051 052 053 054 055
040 041 042 042 043 044 046 047
032 033 034 034 035 036 038 039
024 025 026 026 027 028 030 031
016 017 018 018 019 020 022 023
008 009 010 010 011 012 014 015
000 001 002 003 004 005 006 007
```

Example 3

Function Call:

```
translate(img, 8, 1, 1, 4, 5, 3, 2);
```

Result of print_image:

```
056 057 058 059 060 061 062 063
048 049 050 041 042 043 044 055
040 041 042 033 034 035 036 047
032 033 034 025 026 027 028 039
024 025 026 017 018 019 020 031
016 017 018 009 010 011 012 023
008 009 010 011 012 013 014 015
000 001 002 003 004 005 006 007
```