CS 4323 Design and Implementation of Operating Systems I

Assignment 03: Full Marks 100
(Due Date: 10/14/2019, 11:00 PM CST)

The assignment must be done individually.

In this assignment, you will be using the concepts of multithreading using pthreads. Let us first understand the task and then we will see the implementation details.

How many of you have heard of watermarking or seen the application of watermarking. It is commonly used in the TV channels, where the channel logo is kept mostly in the left top of the screen. This type of watermarking is called visible watermarking, where the channel logo is very much visible to the user's eye. There is invisible watermarking also, where the watermarked image is not visible and is conceived inside the image. Let us briefly understand the concept of watermarking (i.e. visible watermarking, to be specific).

Watermarking is the act of hiding a message related to the signal within the signal itself. The signal can be image, music video or any other video. It is similar to steganography, where both hides a message inside another signal. The difference between them is the message which is being hidden. In the case of steganography, the message to be hidden within a signal has no correlation with the signal, while watermarking tries to hide the message related to the signal within the signal. Watermarking is generally used to identify the ownership of the signal. Consider, if you have produced a masterpiece digital work and if it is copied by somebody, then how would you claim that the content belongs to you? If you have inserted watermark in your masterpiece work, then you can legally claim that the work belongs to you. So, watermarking provides a way to verify the right ownership of any content. It is also used to provide authenticity and integrity of the signal (interested students can explore more about this).

As said earlier, watermarking can be applied to any kind of signal (image, audio, video). However, for this assignment, we will consider the simplest application of watermarking in picture. Any watermarking scheme consists of three main parts:

- Watermark: The message which is to be hidden in the signal
- Encoder: the insertion algorithm used to hide the message in the signal
- Decoder: the extraction algorithm used to extract the watermark from the watermarked signal (i.e. the signal where hides the message)

This assignment will only focus on the encoding scheme. There are many complicated algorithms, but we will focus on the simplest one: Least Significant Bit (LSB) Coding.

Before explaining the encoding algorithm, brief explanation of the image. Below are the two images, we will be using for the assignment.

Fig. Signal (Original Image where watermark is to be inserted)

Fig. watermark (OSU Logo)

Figure 1. Images used for the assignment

Both of these images have a resolution of 250-by-250 pixel. Each pixel contains image information i.e. the intensity value is given as numbers.

Consider smaller images and corelate the concept to the above image:

 This is a grayscale image meaning that each pixel contains 8-bits data. Thus, each pixel contains a value in the range of 0 – 255. This image data is given by:

2	2	3	2	3	3	5	1	4	22	40	44	45	70	60	46	44	29	3	5	9	8	8	10	8
2	3	3	2	4	3	4	53	94	139	178	188	187	186	187	187	185	153	86	40	5	10	9	9	10
2	4	5	3	2	31	111	169	189	184	180	180	179	180	181	181	181	185	190	159	70	10	9	10	11
4	3	4	2	57	162	187	183	179	179	179	181	179	179	180	181	181	181	182	189	179	105	11	9	11
3	4	2	65	185	183	182	179	179	178	178	181	180	183	180	181	183	185	170	152	176	179	71	5	11
4	2	61	178	182	173	166	173	183	184	182	172	170	141	175	184	157	137	155	141	168	187	157	37	6
0	39	161	184	178	177	165	155	112	113	164	172	158	103	161	129	142	160	179	185	181	181	190	89	4
4	97	189	179	180	179	179	178	171	174	160	122	124	107	111	175	180	183	179	180	180	178	182	157	31
35	173	179	177	177	178	179	180	181	180	173	162	167	167	174	145	181	180	179	177	178	179	177	185	44
62	185	178	178	179	179	178	181	179	179	166	138	184	182	174	151	180	179	180	180	178	178	177	183	45
109	182	176	178	179	178	179	179	178	180	179	177	181	179	177	177	160	144	161	175	179	178	178	184	85
136	179	177	177	177	178	178	178	179	179	180	182	171	173	131	199	201	214	189	143	182	178	178	183	101
150	179	176	176	177	177	180	179	179	181	177	153	159	202	171	221	207	211	217	195	175	178	176	183	84
125	179	176	175	176	163	147	176	177	148	184	174	174	220	171	211	197	185	185	175	177	177	176	183	79
87	178	175	175	179	138	174	187	199	150	167	144	130	179	160	206	208	196	200	172	178	177	179	167	37
41	183	174	173	178	127	167	183	216	160	204	191	171	218	178	218	218	197	197	157	179	179	182	138	20
36	176	175	172	177	151	159	161	174	160	181	179	174	190	175	176	171	185	193	169	179	181	170	85	10
10	118	184	172	173	173	167	169	167	175	176	177	178	175	176	172	170	153	117	184	181	179	128	39	1
0	59	171	176	172	174	176	174	176	175	176	177	177	175	177	179	180	178	175	184	180	141	70	5	4
2	5	110	175	171	173	173	175	175	175	174	176	176	176	178	178	178	179	183	175	136	89	18	2	5
3	1	21	132	167	167	175	175	175	174	175	174	175	176	178	179	180	181	164	119	87	23	1	4	5
3	3	1	19	115	149	153	165	169	175	177	179	180	179	178	174	164	127	105	72	16	0	3	6	6
2	3	3	1	9	65	111	125	125	139	146	149	149	148	137	112	92	79	47	3	0	3	6	5	6
2	2	2	2	2	1	19	50	62	75	81	84	84	80	71	55	36	12	1	1	4	7	6	5	6
3	2	2	2	3	3	0	0	3	11	16	17	18	16	10	3	0	1	3	5	5	5	5	5	5

This is a 25-by-25 matrix i.e. number of row = number of column = 25. So, all grayscale image is a 2-D array.

This is true color image i.e. each pixel has red, green and blue component (in short RGB). Each color component (R, G, B) is represented by 8-bits i.e. the range of possible value for each color component is between 0 and 255. This image has the following values:

Red component values:

3	2	2	2	4	6	8	3	9	40	66	72	76	107	95	77	72	47	11	10	13	12	11	13	13
3	3	3	3	6	8	13	76	127	185	230	241	243	243	242	242	236	199	121	62	15	14	12	13	14
2	5	7	7	8	48	147	223	254	255	253	252	251	251	251	250	251	255	254	211	98	21	17	16	16
7	6	8	7	82	212	249	253	252	255	255	255	253	253	253	253	254	255	254	255	234	141	24	17	18
6	9	8	91	236	250	255	254	253	253	254	254	252	253	252	252	253	254	236	216	242	236	100	16	18
6	8	84	231	251	244	238	244	254	255	252	240	234	201	242	252	219	196	214	199	233	252	205	58	11
0	58	209	251	253	250	236	221	168	170	225	231	212	151	219	188	202	224	247	254	252	251	254	123	14
12	130	254	252	254	253	251	245	232	235	221	176	175	156	163	237	246	255	253	254	253	251	252	203	52
58	223	251	251	251	252	253	253	253	251	241	222	228	229	235	206	250	252	253	250	251	251	246	237	74
98	244	252	252	252	253	252	255	252	252	235	199	251	251	241	215	248	246	249	252	250	250	247	238	76
158	244	250	251	252	252	253	252	251	252	249	246	248	244	236	232	210	192	215	238	248	250	250	241	124
191	243	250	251	251	251	251	251	252	252	252	251	230	217	161	220	217	230	215	188	245	250	250	242	146
206	242	249	250	250	248	249	248	250	249	237	200	194	224	184	229	212	215	234	235	235	250	248	241	125
177	241	247	248	246	228	204	229	231	199	224	195	185	227	178	218	202	189	201	215	237	249	248	238	115
129	235	245	248	245	189	207	208	219	171	185	155	137	186	168	213	213	199	216	213	239	250	248	214	58
69	234	242	244	242	176	195	199	232	178	224	211	191	238	198	234	228	201	216	205	244	250	246	176	33
56	220	241	242	242	206	201	195	208	198	221	220	214	228	212	209	195	200	222	227	248	247	225	114	18
19	150	243	240	242	240	229	227	227	236	237	237	237	233	234	228	222	196	162	247	249	238	166	55	4
0	78	217	237	242	246	246	245	248	247	248	248	247	244	247	248	249	241	238	250	241	185	92	12	5
2	14	138	225	234	243	242	243	243	246	246	248	247	245	247	247	247	246	249	235	183	116	27	6	8
4	5	32	166	218	229	241	243	245	245	246	243	245	248	248	248	246	241	215	158	111	33	3	6	7
4	4	4	33	146	194	209	227	234	240	243	242	245	246	240	234	219	171	136	90	23	0	4	8	7
3	5	6	4	18	86	145	170	175	191	198	201	200	197	180	150	125	104	60	6	1	5	9	8	9
4	4	4	4	4	2	30	68	86	104	111	115	115	108	94	72	49	20	2	3	6	11	10	9	9
6	5	3	2	3	4	1	0	8	19	26	29	29	26	17	7	0	2	5	9	9	9	9	9	8

Green component values:

1	2	4	3	3	2	4	0	2	16	31	36	35	63	51	36	35	23	0	3	8	8	8	9	7
1	4	3	2	4	1	0	49	91	137	176	187	185	184	186	186	184	151	81	34	1	9	8	8	9
1	4	4	2	0	27	106	167	188	182	177	177	176	177	179	180	180	183	188	155	66	6	7	8	8
3	2	3	0	52	160	184	179	175	175	175	178	175	176	177	179	178	179	180	188	177	102	6	7	8
2	2	0	60	183	180	178	175	175	175	174	178	177	181	178	178	180	183	167	148	173	177	67	0	9
3	0	56	176	180	169	161	169	181	182	180	169	167	135	172	182	153	131	152	135	165	186	154	32	5
0	35	160	181	175	173	160	151	104	105	161	169	155	96	157	123	137	157	176	182	178	178	188	84	0
1	93	188	176	176	175	175	174	167	170	157	117	122	102	104	172	177	180	176	177	176	175	180	155	25
27	171	175	174	174	174	175	176	178	177	170	159	165	165	171	140	178	176	175	174	175	176	175	183	35
53	182	174	174	176	175	174	178	177	177	163	133	182	180	171	147	178	177	178	178	175	175	174	181	35
104	179	172	175	176	174	176	177	176	178	177	174	179	177	175	176	159	142	159	173	177	175	175	182	80
133	177	174	173	174	175	175	175	176	176	177	180	169	171	130	198	200	212	186	141	180	176	175	180	96
149	177	173	172	174	174	177	176	176	178	174	151	157	200	170	220	206	209	215	193	173	175	173	180	77
121	177	173	172	173	160	143	174	176	145	180	172	173	219	170	210	196	184	183	173	175	174	173	180	74
81	177	172	171	175	134	171	185	199	149	165	142	128	177	158	205	208	196	199	170	175	174	176	166	32
32	181	172	170	174	123	163	181	215	159	202	189	169	217	177	217	217	196	196	156	176	176	180	137	16
29	174	173	170	174	147	155	158	172	159	179	177	172	189	174	175	169	183	192	168	178	179	168	83	7
7	116	182	170	170	170	164	166	164	172	173	176	177	174	175	171	169	151	113	183	180	177	126	36	0
0	56	169	173	169	170	174	172	173	172	173	175	175	174	176	178	179	176	173	182	178	139	67	3	4
2	2	107	173	168	170	171	173	173	172	171	173	173	175	176	176	177	177	181	173	133	86	16	1	4
3	0	17	129	164	164	172	173	173	171	173	173	173	173	176	178	179	179	161	117	85	21	0	4	5
3	3	0	15	113	147	150	161	166	172	175	178	178	176	176	173	163	124	102	70	15	0	3	6	6
2	3	2	0	7	63	108	121	120	135	143	145	146	145	136	110	89	76	44	2	0	3	5	4	5
1	1	1	2	1	0	17	46	59	72	78	81	82	78	70	53	33	10	0	0	4	5	4	4	5
1	1	1	2	3	3	0	0	1	8	12	13	14	12	8	2	0	0	2	4	4	3	3	4	4

Blue component values:

2	1	0	0	2	2	2	0	0	5	15	16	13	13	13	14	16	9	0	1	2	1	4	4	3
5	2	0	0	1	0	0	10	24	32	55	57	51	48	49	52	57	45	24	10	0	1	3	6	6
4	4	1	0	0	3	38	38	27	12	6	8	6	5	6	6	6	10	29	46	15	0	2	6	9
1	2	1	0	20	42	36	19	5	0	0	4	2	0	1	3	2	1	6	25	43	25	0	2	6
0	0	0	25	61	21	11	7	3	0	1	7	11	13	5	7	14	18	11	7	17	42	15	0	4
2	0	26	46	14	8	6	9	10	8	10	11	21	17	12	15	14	13	13	16	12	26	46	8	1
1	6	41	20	0	3	1	2	5	4	17	34	33	13	33	4	9	9	14	19	11	10	31	25	0
0	28	27	3	4	3	8	19	29	32	18	10	4	2	14	32	20	7	1	4	7	4	12	51	11
12	53	11	1	2	4	4	7	10	10	12	21	17	18	30	10	14	8	5	0	4	4	6	59	16
13	47	7	2	5	4	2	2	0	2	0	5	15	11	13	3	13	16	12	4	2	3	6	50	14
9	33	5	2	4	4	2	0	1	3	4	10	16	22	31	41	35	25	30	20	11	3	8	46	12
6	21	2	0	1	3	5	3	3	4	6	15	30	66	60	150	163	181	132	35	29	3	8	42	11
6	20	2	0	2	6	15	13	7	20	38	43	79	151	141	204	198	207	182	102	29	1	6	43	11
7	26	4	0	9	11	14	45	39	27	96	129	152	207	156	198	188	178	149	84	33	2	6	54	14
10	36	4	2	27	22	103	138	148	103	131	123	118	167	146	192	198	189	162	78	32	2	10	52	12
15	56	5	0	28	19	110	152	179	117	158	148	128	175	133	183	200	189	149	40	21	5	24	43	6
18	72	13	0	20	25	73	90	91	69	86	83	78	96	81	99	120	153	121	23	7	16	38	20	1
0	43	42	8	6	17	24	30	23	28	28	26	29	27	27	27	42	49	23	20	10	35	41	14	0
0	26	64	28	5	2	3	1	1	0	1	3	2	1	3	2	7	19	18	19	33	37	29	0	2
1	0	50	55	18	7	3	4	5	2	0	4	3	2	6	5	4	10	23	30	32	33	6	0	5
1	0	9	56	47	22	15	8	5	0	0	0	0	1	3	4	11	34	44	29	32	11	0	2	4
1	1	0	7	45	42	23	21	17	18	17	19	22	22	22	25	29	24	37	35	7	0	1	4	4
0	1	1	0	0	22	34	28	21	26	27	29	29	32	31	21	20	31	24	0	0	1	2	3	4
1	2	2	1	0	0	1	20	17	15	14	17	17	17	17	17	18	3	0	0	2	5	3	3	4
3	3	3	0	0	1	1	0	0	3	6	7	8	6	4	1	0	3	2	2	3	3	5	4	3

This is a 25-by-25-by-3 matrix i.e. number of row = number of column = 25 and you have three color components (Red, Blue and Green. In short, RGB). So, all true color image is a 3-D array.

The left image in figure 1 is a true color image of size 250-by-250-by-3 while the right image in figure 1 is a gray color image of size 250-by-250. You are given 4 txt files along with this assignment:

- redComponent.txt
- greenComponent.txt
- blueComponent.txt
- OSU.txt (gray color image)

The first three .txt files represent the three color components (R, G and B) for the left image (scenery image). The last .txt file represent the gray color component for the right image (OSU logo).

All together you have to use 4 threads: three threads (one thread per each color component) to process the true color image and one thread to perform task on the grayscale image. The steps to be followed by each thread are as follows:

1. Each thread should read the data from the corresponding color component .txt file.
   - First three threads perform read for the truecolor image
   - The last thread (i.e. 4th thread) performs read for the gray color image

2. The first three threads now removes the least significant bits (LSB) from each cell in their matrix i.e. if the LSB is 0 then it remains 0 but if the LSB is 1 then it is converted to 0.

For e.g.: consider the red component of the true color image as:

129	46	180	127	53
12	35	245	255	24
9	0	212	123	63
23	45	190	35	145
1	46	32	26	150

This matrix gets changed into:

128	46	180	126	52
12	34	244	254	24
8	0	212	122	62
22	44	190	34	144
1	46	32	26	150

All the 3 threads perform this operation for their respective matrix.

3. The 4th thread needs to convert the corresponding 2-D matrix (i.e. gray color image data) into binary form i.e. the values in this matrix should be either zero or one (this is black & white image). The way to do this is:
    - If any cell value is between 0 and 127 (inclusive), then make it to 0.
    - Remaining values i.e. in the range between 128 and 255 (inclusive), it is converted into 1.

    For e.g.: Given the matrix (i.e. gray color image data):

129	46	180	127	53
12	35	245	255	24
9	0	212	123	63
23	45	190	35	145
1	46	32	26	150

    The corresponding black-and-white image data is:

1	0	1	0	0
0	0	1	1	0
0	0	1	0	0
0	0	1	0	1
0	0	0	0	1

    The 4th thread's task end here.

4. Each 3 threads (i.e. first 3 threads) need to add the corresponding matrices obtained from step 2 with the matrix obtained from step 3. It needs to be ensured that the minimum

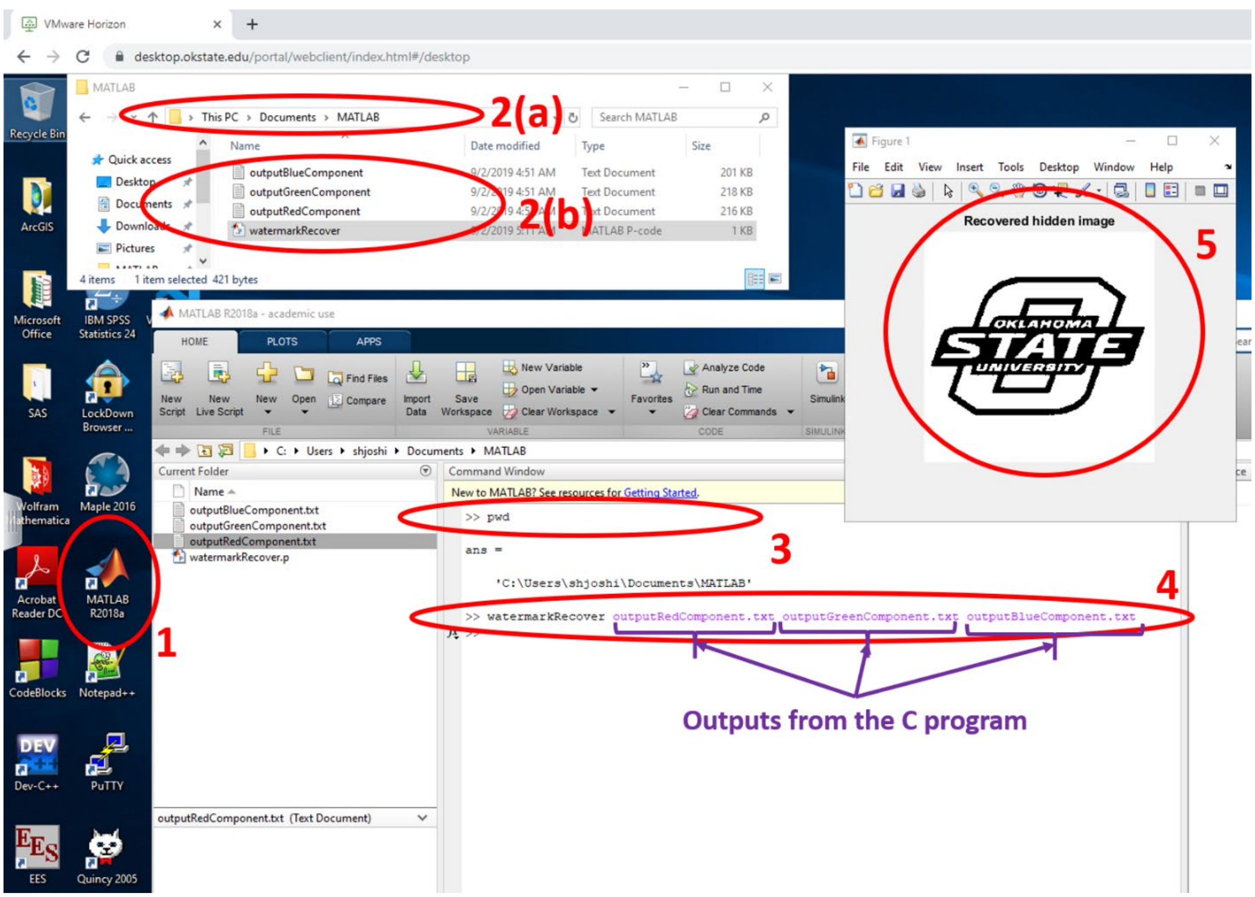
value can be only 0 and maximum value in any cell can be 255. Any value lesser than 0 has to be converted to 0 and any value larger than 255 has to be converted into 255.

5. Each thread needs to save the final matrix in the .txt file as:
    - outputRedComponent.txt for red component
    - outputGreenComponent.txt for green component
    - outputBlueComponent.txt for blue component

This is it. You have successfully inserted the watermark in the image.

<u>The following in not the part of the assignment.</u> If anyone is interested to check if their final outputs are correct or not then they can check by following the steps given below:

- Visit the okstate virtual lab at: https://desktop.okstate.edu/portal/webclient/index.html#/
- Run the MATLAB
- Along with this assignment, you have been given watermarkRecover.p file. Copy this .p file along with your outputs from your c file into Documents > MATLAB folder.
- In the command window of MATLAB, type:
  watermarkRecover outputRedComponent.txt outputGreenComponent.txt outputBlueComponent.txt



If you see output (as shown above, by number 5), then your program is correct.

Submission Guidelines:

- You should submit your programming assignment as a single .c file: Assignment03_LastName_FirstName_01.c (where ** means assignment number and XX means question number). Example: Assignment03_Andrew_Simon_01.c

- Your code should include the header information, which should include:
    - Name
    - CWID
    - Email
    - Date

  followed by a brief description of the program.