Name \_\_\_\_\_ Chen，Yanni \_\_\_\_\_\_\_ Date: \_\_\_12/21/2020\_\_\_ (last name, first name)

Section: \_\_\_\_\_005\_\_\_\_\_\_

**Final Exam Report**

**Total in points** (100 points total): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **Professor’s Comments:**

**1.**

Minus zero 10000000 0 -2 -0.0

- 01000101 21/16 1 2.625

Smallest denorm- 10001111 15/16 -2 -0.234375

alized(negative)

Largest normaliz- 01101111 31/16 3 15.5

ed(positive)

One 00110000 1 0 1.0

- 01010110 11/8 2 5.5

Positive infinity 01110000 - - +∞

**2.**

**A.** f[1] **B.** b.i->f[3] **C.** i->e **D.** i->g->d->a[1]

**3.**

**3.1**

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| c | //////////////////////////// | value | next | flag | left | right |

**3.2**

NONE NONE E NONE

**4.**

|  |  |  |
| --- | --- | --- |
| Version | Measured CPE | Theoretical CPE |
| A1 | 4.00 | 12/3=4.00 |
| A2 | 2.67 | 8/3=2.67 |
| A3 | 1.67 | 4/3=1.33 |
| A4 | 1.67 | 4/3=1.33 |
| A5 | 2.67 | 8/3=2.67 |

**5.**

**5.1**

100% 25% 100%

**5.2**

25% 25% 25% 75%

**6.**

**6.1**

**A.**

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

|  |  |  |
| --- | --- | --- |
| TLBT | TLBI |  |

VPN VPO

**B.**

12 11 10 9 8 7 6 5 4 3 2 1 0

|  |  |
| --- | --- |
| PPN | PPO |

CT CI CO

**6.2**

**A.** 0001110111011110

**B.** 0x0E 0x00 0x07 N N 0x01

**C.** 0001111011110

**D.** 0x02 0x07 0x1E Y 0x0F

**7-8.**

char rotate\_descr[] = "rotate: Current working version";

void rotate(int dim, pixel \*src, pixel \*dst)

{

int i, j;

if (src==dst )

{

naive\_rotate(dim, src, dst);

return;

}

for (i = 0; i < dim; i+=4)

for (j = 0; j < dim; j+=4)

{

dst[RIDX(dim-1-j, i, dim)] = src[RIDX(i, j, dim)];

dst[RIDX(dim-2-j, i, dim)] = src[RIDX(i, j+1, dim)];

dst[RIDX(dim-3-j, i, dim)] = src[RIDX(i, j+2, dim)];

dst[RIDX(dim-4-j, i, dim)] = src[RIDX(i, j+3, dim)];

dst[RIDX(dim-4-j, i+1, dim)] = src[RIDX(i+1, j+3, dim)];

dst[RIDX(dim-3-j, i+1, dim)] = src[RIDX(i+1, j+2, dim)];

dst[RIDX(dim-2-j, i+1, dim)] = src[RIDX(i+1, j+1, dim)];

dst[RIDX(dim-1-j, i+1, dim)] = src[RIDX(i+1, j, dim)];

dst[RIDX(dim-1-j, i+2, dim)] = src[RIDX(i+2, j, dim)];

dst[RIDX(dim-2-j, i+2, dim)] = src[RIDX(i+2, j+1, dim)];

dst[RIDX(dim-3-j, i+2, dim)] = src[RIDX(i+2, j+2, dim)];

dst[RIDX(dim-4-j, i+2, dim)] = src[RIDX(i+2, j+3, dim)];

dst[RIDX(dim-4-j, i+3, dim)] = src[RIDX(i+3, j+3, dim)];

dst[RIDX(dim-3-j, i+3, dim)] = src[RIDX(i+3, j+2, dim)];

dst[RIDX(dim-2-j, i+3, dim)] = src[RIDX(i+3, j+1, dim)];

dst[RIDX(dim-1-j, i+3, dim)] = src[RIDX(i+3, j, dim)];

}

}

char rotate1\_descr[] = "rotate: Additional version 1";

void rotate1(int dim, pixel \*src, pixel \*dst)

{

int i, j;

if (src==dst )

{

naive\_rotate(dim, src, dst);

return;

}

for (i = 0; i < dim; i+=2)

for (j = 0; j < dim; j+=4)

{

dst[RIDX(dim-1-j, i, dim)] = src[RIDX(i, j, dim)];

dst[RIDX(dim-2-j, i, dim)] = src[RIDX(i, j+1, dim)];

dst[RIDX(dim-3-j, i, dim)] = src[RIDX(i, j+2, dim)];

dst[RIDX(dim-4-j, i, dim)] = src[RIDX(i, j+3, dim)];

dst[RIDX(dim-4-j, i+1, dim)] = src[RIDX(i+1, j+3, dim)];

dst[RIDX(dim-3-j, i+1, dim)] = src[RIDX(i+1, j+2, dim)];

dst[RIDX(dim-2-j, i+1, dim)] = src[RIDX(i+1, j+1, dim)];

dst[RIDX(dim-1-j, i+1, dim)] = src[RIDX(i+1, j, dim)];

}

}

char rotate2\_descr[] = "rotate: Additional version 2";

void rotate2(int dim, pixel \*src, pixel \*dst)

{

int i, j;

if (src==dst )

{

naive\_rotate(dim, src, dst);

return;

}

for (i = 0; i < dim; i+=2)

for (j = 0; j < dim; j+=2)

{

dst[RIDX(dim-1-j, i, dim)] = src[RIDX(i, j, dim)];

dst[RIDX(dim-2-j, i, dim)] = src[RIDX(i, j+1, dim)];

dst[RIDX(dim-2-j, i+1, dim)] = src[RIDX(i+1, j+1, dim)];

dst[RIDX(dim-1-j, i+1, dim)] = src[RIDX(i+1, j, dim)];

}

}

**9-10．**

char smooth\_descr[] = "smooth: Current working version";

void smooth(int dim, pixel \*src, pixel \*dst)

{

int i, j;

if (src==dst )

{

naive\_smooth(dim, src, dst);

return;

}

dst[RIDX(0, 0, dim)].red = (src[RIDX(0, 0, dim)].red+src[RIDX(0, 1, dim)].red+src[RIDX(1, 1, dim)].red+src[RIDX(1, 0, dim)].red)/4;

dst[RIDX(0, 0, dim)].green = (src[RIDX(0, 0, dim)].green+src[RIDX(0, 1, dim)].green+src[RIDX(1, 1, dim)].green+src[RIDX(1, 0, dim)].green)/4;

dst[RIDX(0, 0, dim)].blue = (src[RIDX(0, 0, dim)].blue+src[RIDX(0, 1, dim)].blue+src[RIDX(1, 1, dim)].blue+src[RIDX(1, 0, dim)].blue)/4;

dst[RIDX(0, dim-1, dim)].red = (src[RIDX(0, dim-1, dim)].red+src[RIDX(1, dim-1, dim)].red+src[RIDX(1, dim-2, dim)].red+src[RIDX(0, dim-2, dim)].red)/4;

dst[RIDX(0, dim-1, dim)].green = (src[RIDX(0, dim-1, dim)].green+src[RIDX(1, dim-1, dim)].green+src[RIDX(1, dim-2, dim)].green+src[RIDX(0, dim-2, dim)].green)/4;

dst[RIDX(0, dim-1, dim)].blue = (src[RIDX(0, dim-1, dim)].blue+src[RIDX(1, dim-1, dim)].blue+src[RIDX(1, dim-2, dim)].blue+src[RIDX(0, dim-2, dim)].blue)/4;

dst[RIDX(dim-1, 0, dim)].red = (src[RIDX(dim-1, 0, dim)].red+src[RIDX(dim-2, 0, dim)].red+src[RIDX(dim-2, 1, dim)].red+src[RIDX(dim-1, 1, dim)].red)/4;

dst[RIDX(dim-1, 0, dim)].green = (src[RIDX(dim-1, 0, dim)].green+src[RIDX(dim-2, 0, dim)].green+src[RIDX(dim-2, 1, dim)].green+src[RIDX(dim-1, 1, dim)].green)/4;

dst[RIDX(dim-1, 0, dim)].blue = (src[RIDX(dim-1, 0, dim)].blue+src[RIDX(dim-2, 0, dim)].blue+src[RIDX(dim-2, 1, dim)].blue+src[RIDX(dim-1, 1, dim)].blue)/4;

dst[RIDX(dim-1, dim-1, dim)].red = (src[RIDX(dim-1, dim-1, dim)].red+src[RIDX(dim-1, dim-2, dim)].red+src[RIDX(dim-2, dim-2, dim)].red+src[RIDX(dim-2, dim-1, dim)].red)/4;

dst[RIDX(dim-1, dim-1, dim)].green = (src[RIDX(dim-1, dim-1, dim)].green+src[RIDX(dim-1, dim-2, dim)].green+src[RIDX(dim-2, dim-2, dim)].green+src[RIDX(dim-2, dim-1, dim)].green)/4;

dst[RIDX(dim-1, dim-1, dim)].blue = (src[RIDX(dim-1, dim-1, dim)].blue+src[RIDX(dim-1, dim-2, dim)].blue+src[RIDX(dim-2, dim-2, dim)].blue+src[RIDX(dim-2, dim-1, dim)].blue)/4;

for (i = 1; i < dim-1; i++)

{

dst[RIDX(i, 0, dim)].red = (src[RIDX(i-1, 0, dim)].red+src[RIDX(i-1, 1, dim)].red+src[RIDX(i, 1, dim)].red+src[RIDX(i, 0, dim)].red+src[RIDX(i+1, 0, dim)].red+src[RIDX(i+1, 1, dim)].red)/6;

dst[RIDX(i, 0, dim)].green = (src[RIDX(i-1, 0, dim)].green+src[RIDX(i-1, 1, dim)].green+src[RIDX(i, 1, dim)].green+src[RIDX(i, 0, dim)].green+src[RIDX(i+1, 0, dim)].green+src[RIDX(i+1, 1, dim)].green)/6;

dst[RIDX(i, 0, dim)].blue = (src[RIDX(i-1, 0, dim)].blue+src[RIDX(i-1, 1, dim)].blue+src[RIDX(i, 1, dim)].blue+src[RIDX(i, 0, dim)].blue+src[RIDX(i+1, 0, dim)].blue+src[RIDX(i+1, 1, dim)].blue)/6;

dst[RIDX(i, dim-1, dim)].red = (src[RIDX(i-1, dim-1, dim)].red+src[RIDX(i-1, dim-2, dim)].red+src[RIDX(i, dim-2, dim)].red+src[RIDX(i, dim-1, dim)].red+src[RIDX(i+1, dim-1, dim)].red+src[RIDX(i+1, dim-2, dim)].red)/6;

dst[RIDX(i, dim-1, dim)].green = (src[RIDX(i-1, dim-1, dim)].green+src[RIDX(i-1, dim-2, dim)].green+src[RIDX(i, dim-2, dim)].green+src[RIDX(i, dim-1, dim)].green+src[RIDX(i+1, dim-1, dim)].green+src[RIDX(i+1, dim-2, dim)].green)/6;

dst[RIDX(i, dim-1, dim)].blue = (src[RIDX(i-1, dim-1, dim)].blue+src[RIDX(i-1, dim-2, dim)].blue+src[RIDX(i, dim-2, dim)].blue+src[RIDX(i, dim-1, dim)].blue+src[RIDX(i+1, dim-1, dim)].blue+src[RIDX(i+1, dim-2, dim)].blue)/6;

dst[RIDX(0, i, dim)].red = (src[RIDX(0, i-1, dim)].red+src[RIDX(0, i, dim)].red+src[RIDX(0, i+1, dim)].red+src[RIDX(1, i+1, dim)].red+src[RIDX(1, i, dim)].red+src[RIDX(1, i-1, dim)].red)/6;

dst[RIDX(0, i, dim)].green = (src[RIDX(0, i-1, dim)].green+src[RIDX(0, i, dim)].green+src[RIDX(0, i+1, dim)].green+src[RIDX(1, i+1, dim)].green+src[RIDX(1, i, dim)].green+src[RIDX(1, i-1, dim)].green)/6;

dst[RIDX(0, i, dim)].blue = (src[RIDX(0, i-1, dim)].blue+src[RIDX(0, i, dim)].blue+src[RIDX(0, i+1, dim)].blue+src[RIDX(1, i+1, dim)].blue+src[RIDX(1, i, dim)].blue+src[RIDX(1, i-1, dim)].blue)/6;

dst[RIDX(dim-1, i, dim)].red = (src[RIDX(dim-1, i-1, dim)].red+src[RIDX(dim-1, i, dim)].red+src[RIDX(dim-1, i+1, dim)].red+src[RIDX(dim-2, i+1, dim)].red+src[RIDX(dim-2, i, dim)].red+src[RIDX(dim-2, i-1, dim)].red)/6;

dst[RIDX(dim-1, i, dim)].green = (src[RIDX(dim-1, i-1, dim)].green+src[RIDX(dim-1, i, dim)].green+src[RIDX(dim-1, i+1, dim)].green+src[RIDX(dim-2, i+1, dim)].green+src[RIDX(dim-2, i, dim)].green+src[RIDX(dim-2, i-1, dim)].green)/6;

dst[RIDX(dim-1, i, dim)].blue = (src[RIDX(dim-1, i-1, dim)].blue+src[RIDX(dim-1, i, dim)].blue+src[RIDX(dim-1, i+1, dim)].blue+src[RIDX(dim-2, i+1, dim)].blue+src[RIDX(dim-2, i, dim)].blue+src[RIDX(dim-2, i-1, dim)].blue)/6;

for (j = 1; j < dim-1; j++)

{

dst[RIDX(i, j, dim)].red = (src[RIDX(i-1, j-1, dim)].red+src[RIDX(i-1, j, dim)].red+src[RIDX(i-1, j+1, dim)].red+src[RIDX(i, j+1, dim)].red+src[RIDX(i, j, dim)].red+src[RIDX(i, j-1, dim)].red+src[RIDX(i+1, j-1, dim)].red+src[RIDX(i+1, j, dim)].red+src[RIDX(i+1, j+1, dim)].red)/9;

dst[RIDX(i, j, dim)].green = (src[RIDX(i-1, j-1, dim)].green+src[RIDX(i-1, j, dim)].green+src[RIDX(i-1, j+1, dim)].green+src[RIDX(i, j+1, dim)].green+src[RIDX(i, j, dim)].green+src[RIDX(i, j-1, dim)].green+src[RIDX(i+1, j-1, dim)].green+src[RIDX(i+1, j, dim)].green+src[RIDX(i+1, j+1, dim)].green)/9;

dst[RIDX(i, j, dim)].blue = (src[RIDX(i-1, j-1, dim)].blue+src[RIDX(i-1, j, dim)].blue+src[RIDX(i-1, j+1, dim)].blue+src[RIDX(i, j+1, dim)].blue+src[RIDX(i, j, dim)].blue+src[RIDX(i, j-1, dim)].blue+src[RIDX(i+1, j-1, dim)].blue+src[RIDX(i+1, j, dim)].blue+src[RIDX(i+1, j+1, dim)].blue)/9;

}

}

}

char smooth1\_descr[] = "smooth: Additional version 1";

void smooth1(int dim, pixel \*src, pixel \*dst)

{

int i, j;

if (src==dst )

{

naive\_smooth(dim, src, dst);

return;

}

dst[RIDX(0, 0, dim)] = avg(dim, 0, 0, src);

dst[RIDX(0, dim-1, dim)] = avg(dim, 0, dim-1, src);

dst[RIDX(dim-1, 0, dim)] = avg(dim, dim-1, 0, src);

dst[RIDX(dim-1, dim-1, dim)] = avg(dim, dim-1, dim-1, src);

for (i = 1; i < dim-1; i++)

{

dst[RIDX(i, 0, dim)] = avg(dim, i, 0, src);

dst[RIDX(i, dim-1, dim)] = avg(dim, i, dim-1, src);

dst[RIDX(0, i, dim)] = avg(dim, 0, i, src);

dst[RIDX(dim-1, i, dim)] = avg(dim, dim-1, i, src);

for (j = 1; j < dim-1; j++)

{

dst[RIDX(i, j, dim)].red = (src[RIDX(i-1, j-1, dim)].red+src[RIDX(i-1, j, dim)].red+src[RIDX(i-1, j+1, dim)].red+src[RIDX(i, j+1, dim)].red+src[RIDX(i, j, dim)].red+src[RIDX(i, j-1, dim)].red+src[RIDX(i+1, j-1, dim)].red+src[RIDX(i+1, j, dim)].red+src[RIDX(i+1, j+1, dim)].red)/9;

dst[RIDX(i, j, dim)].green = (src[RIDX(i-1, j-1, dim)].green+src[RIDX(i-1, j, dim)].green+src[RIDX(i-1, j+1, dim)].green+src[RIDX(i, j+1, dim)].green+src[RIDX(i, j, dim)].green+src[RIDX(i, j-1, dim)].green+src[RIDX(i+1, j-1, dim)].green+src[RIDX(i+1, j, dim)].green+src[RIDX(i+1, j+1, dim)].green)/9;

dst[RIDX(i, j, dim)].blue = (src[RIDX(i-1, j-1, dim)].blue+src[RIDX(i-1, j, dim)].blue+src[RIDX(i-1, j+1, dim)].blue+src[RIDX(i, j+1, dim)].blue+src[RIDX(i, j, dim)].blue+src[RIDX(i, j-1, dim)].blue+src[RIDX(i+1, j-1, dim)].blue+src[RIDX(i+1, j, dim)].blue+src[RIDX(i+1, j+1, dim)].blue)/9;

}

}

}

char smooth2\_descr[] = "smooth: Additional version 2";

void smooth2(int dim, pixel \*src, pixel \*dst)

{

int i, j;

if (src==dst )

{

naive\_smooth(dim, src, dst);

return;

}

for (i = 0; i < dim; i+=4)

for (j = 0; j < dim; j+=4)

{

dst[RIDX(i, j, dim)] = avg(dim, i, j, src);

dst[RIDX(i, j+1, dim)] = avg(dim, i, j+1, src);

dst[RIDX(i, j+2, dim)] = avg(dim, i, j+2, src);

dst[RIDX(i, j+3, dim)] = avg(dim, i, j+3, src);

dst[RIDX(i+1, j+3, dim)] = avg(dim, i+1, j+3, src);

dst[RIDX(i+1, j+2, dim)] = avg(dim, i+1, j+2, src);

dst[RIDX(i+1, j+1, dim)] = avg(dim, i+1, j+1, src);

dst[RIDX(i+1, j, dim)] = avg(dim, i+1, j, src);

dst[RIDX(i+2, j, dim)] = avg(dim, i+2, j, src);

dst[RIDX(i+2, j+1, dim)] = avg(dim, i+2, j+1, src);

dst[RIDX(i+2, j+2, dim)] = avg(dim, i+2, j+2, src);

dst[RIDX(i+2, j+3, dim)] = avg(dim, i+2, j+3, src);

dst[RIDX(i+3, j+3, dim)] = avg(dim, i+3, j+3, src);

dst[RIDX(i+3, j+2, dim)] = avg(dim, i+3, j+2, src);

dst[RIDX(i+3, j+1, dim)] = avg(dim, i+3, j+1, src);

dst[RIDX(i+3, j, dim)] = avg(dim, i+3, j, src);

}

}