Lab Exercise
CPE 357
Systems Programming

CAL POLY
SAN LUIS OBISPO

LAB 3

Exercise:

Lets speed up working on a file:
Read any bitmap file and rotate the image around its middle point. However, do this in **N** (1-4) processes to speed up the CPU worktime.
Measure the time print it.

Howto:

Similar to program 1, read a bitmap (BMP) file into memory. Allocate your memory with mmap() and don't forget to set the shared memory flag.
Fork the process with fork() so that **N** processes work on the file at the same time.

Program call:
./yourprogram [IMAGEFILE] [ROTATING ANGLE] [N PROCESSES] [OUTPUTFILE]
[IMAGEFILE] that's your bitmap
[ROTATING ANGLE] a floating point number in radiants
[N PROCESSES] number of processes involved. 1-4.
[OUTPUTFILE] the output file

Example:
./rotate lion.bmp 2.1 4 result.bmp

Submission:

In person

For the rotation:

Fear not, it's not that hard. Here is the receipt:

- Go over every pixel in x and y. Assume this is already the end state.
- Subtract the middle point from these coordinates e.g. x_r = x – w/2; same for y
- Rotate with a rotation matrix calculation, doesn't matter how you do this, but here is the formula, but take the inverse alpha (-alpha):

$$\begin{pmatrix} x\_ot \\ y\_ot \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x\_r \\ y\_r \end{pmatrix}$$

- Now add the middle coords back: x_o = x_ot + w/2; same for y.
- Now you have the original coordinates x_o and y_o. Check if they are within the boundaries, if not, put black into the pixel at x and y. But if, then sample the color at x_o and y_o into x and y.

HINTS:

As always with programs taking command line arguments, do NOT start with taking command line args. Hardcode them. You will need your debugger!
This program requires multiple CPU cores.
It's certain your laptop/desktop has this, but you need to make sure to activate these resources in your virtual solution. A server, like the Cal Poly server, WILL NOT let you take more than one core.

RESULT: