Name: _____

Date: _____

Collaborators: _____

(If applicable, the collaborators submit their individually written assignments together)

Question:	1	2	3	4	5	Total
Points:	5	30	30	5	5	75
Score:						

Instructor/grader comments:

1. (5 points) Project initialization:

   ssh to your virtual machine (simultaneously creating an ssh tunnel that connects ports 8888 on both ends).

   Once per assignment: create a directory for this homework assignment (say **hw08**).

   Once per assignment: from the command prompt start julia, switch to the package mode and activate your project. Install the package `IJulia` locally - that is needed to create Project.toml and Manifest.toml project files. (You can add other required packages when working with notebooks.)

   Exit julia and launch a jupyter server on the virtual machine, e.g.

   ```
   julia -e 'using IJulia; notebook(dir=".",detached=true)'
   ```

Working with code. Image processing

2. (30 points) Write a julia script that downloads the image https://www.phys.uconn.edu/~rozman/Courses/Eye-Chart-Template-03.jpg and carves 430 top-to-bottom seams and 380 left-to-right seams. Your code must display only three images: the original one; the result of carving top-to-bottom seams; the final image.

   Start from the code for seam carving discussed in class, remove all unneeded code components, and adapt the code to carving both top-to-bottom and left-to-right seams.

   Place your code for Problem 2 into a notebook **hw08p2-seam-carving.ipynb**

Monte Carlo methods

3. Consider the Monte Carlo algorithm for calculating π that was discussed in class.

   (a) Use the following Julia function, `mcpi(np)`, that accepts the number of trials *np*, and implement the Monte Carlo method discussed in class.

   ```
   function mcpi(np)
       s = 0
       for i = 1:np
           x = rand()
           y = rand()
           if x^2 + y^2 <= 1.0
               s += 1
           end
       end
       return 4*s/np
   end
   ```

(b) (15 points) Use the following code fragment to determine the convergence rate of the algorithm and the errors.

```
n = 14
pis = zeros(n)
pistds = zeros(n)
np = zeros(Int64, n)
k = 256
res = zeros(k)
ns = ones(Int64, k)
for i = 1:n
    @show i
    np[i] = 2^(i+10)
    res .= map(mcpi, np[i]*ns)
    pis[i] = mean(res)
    pistds[i] = std(res, mean=pis[i])/sqrt(k)
end
```

Seed your random number generator with a seed of your choice.

Plot your results as follows.

```
loglog(np, pistds, "o", label="Std/sqrt(k)")
loglog(np, abs.(pis .- pi), "o", label="Err")
loglog(np, 0.25 ./ sqrt.(np), "--", label=L"$1/\sqrt{n}$")
grid(true)
xlabel("MC steps")
legend()
```

Describe and explain your observation in your README.md file.

(c) (15 points) Convert your code for calculations on 4 CPUs.

To protect from accidentally over-adding workers, add processes as follows:

```
if nprocs() == 1
    addprocs(4)
end
```

Optional (but recommended), run the following code immediately after adding workers:

```
@everywhere begin
    using Pkg
    Pkg.activate(".")
    Pkg.instantiate()
end
```

Seed the random number generator on every worker with a different seed, e.g.

```
@everywhere begin
    seed = myid()
    Random.seed!(seed)
end
```

Submit for grading only the distributed version of your code.

Place your code for Problem 3 into a notebook **hw08p3-mc-pi.ipynb**

Git

4. (5 points)

   Clean the cells of your jupyter notebook(s).

   Create a git repository for hw08. Check your notebook(s), Julia project files (Project.toml and Manifest.toml), and .gitignore file into the repository. Provide meaningful commit messages.

Gitlab

5. (5 points)

   Create an empty Gitlab project called **hw08** (name it exactly as shown). At this step un-check the box "Create README.md file".

   Push the content of your git repository to Gitlab's **hw08** project

   On the Gitlab "side" create README.md file.

   Pull the README.md file to your local git repository to synchronize your local and remote repositories.

   Share the project with the instructor and grant him **Reporter** privileges.