# Introduction: Hieroglyphic Numeral System of the Ancient Egyptians

This homework will require that you use computer vision techniques on some input images in order to find and interpret the following ancient Egyptian hieroglyphs:

| 1 | 10 | 100 | 1,000 | 10,000 | 100,000 | 1,000,000 | Addition | Subtraction |
|---|---|---|---|---|---|---|---|---|
|  133FA |  13386 |  13362 |  131BC |  130AD |  13190 |  1304F |  130BD |  130BB |

The following links should sufficiently describe how to interpret the ancient Egyptian numeral system:

Saint Louis University [http://euler.slu.edu/escher/index.php/History_and_Numbers]

Wikipedia [http://en.wikipedia.org/wiki/Egyptian_numerals]

For this assignment, you will need to create a wiki page to describe your solutions to the given tasks. Your wiki is meant to exhibit your programs and the images which they produce. Save images in a web-compatible format (ie. jpeg, png, etc.) and make sure that your code formatting is correctly preserved when you finally put it up on your wiki. You can use this page's markup to build your own wiki, but word of caution: don't put off the wiki formatting process for too long.

You should also consider qualifying your submission with useful or interesting descriptions about, for instance, how your code works or the development process which led you to your solution.

The supplied zip file contains all of the input images given below (as well as some other resources which you may find useful). You should expect that we will test your programs against similar input images, and that we will be reading your code; don't write a solution which can only solve the given input images. For example, when coloring hieroglyphs, don't write code which just performs flood-fill on a few convenient hard-coded pixels!

Note that any directions about how to solve these problems are only suggestions. If you think up a completely different strategy to solve the given task, go for it. Many artificial constraints have been placed upon the given inputs so that the implementer can make some very important simplifying assumptions. If you are looking to develop more robust hieroglyph-interpreting CV solutions, go for it! Just be warned, generalization of CV problems can get very complicated very quickly.

# Part 1: Basic Morphological Operators

For each of the tasks in Part 1 of this assignment, you will need to produce two working programs: one written in MATLAB and another written in either C or C++. Both programs should independently solve the following task.

Using morphological operators and simple bit-masks, separate each of the given images into three segments as shown below. Use the segmented images to produce another image which clearly distinguishes these three regions with distinct colors.

## Task A: Paint By Numbers

In this task, divide the hieroglyph up into three segments: a) the stem, b) the top, and c) the base. Using these segments, create an image like d) below.

| Input | a) Stem | b) Top | c) Base | d) Colored |
|---|---|---|---|---|
|  |  |  |  |  |

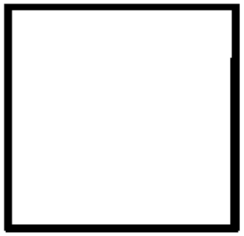
### Code

```
Insert MATLAB code here.
```

```
Insert C/C++ code here.
```

# Task B: Free The Falcon

In this task, divide the hieroglyph up into three segments: a) the outer perimeter, b) the top right corner, and c) the bird icon inside the box. Using these segments, create an image like d) below.

| Input | a) Perimeter | b) Corner | c) Bird | d) Colored |
|---|---|---|---|---|
|  |  |  |  |  |

## Code

```
Insert MATLAB code here.
```

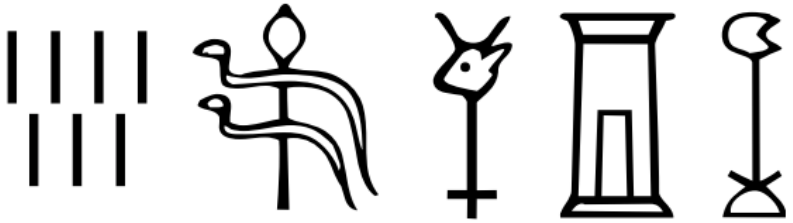
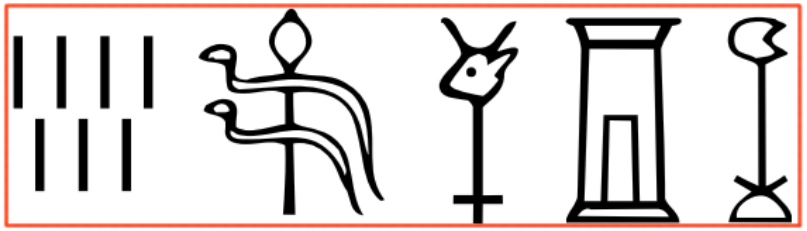
```
Insert C/C++ code here.
```

# Part 2: Horizontal/Vertical Projection

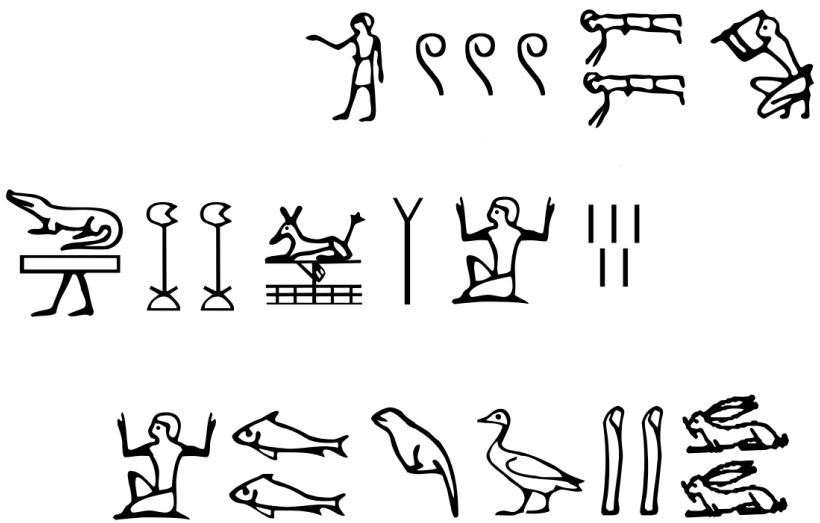
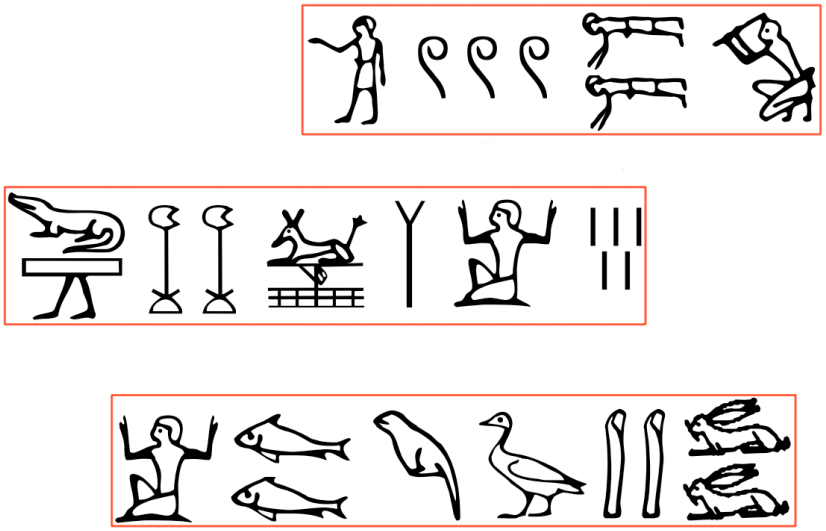
For Part 2 of this assignment, you will need to produce two working programs: one written in MATLAB and another written in either C or C++. Both programs should be able to solve both Task A and Task B.

Notice that the hieroglyphs in both of the input images are aligned in horizontal rows. Use projection to find and create bounding boxes around each of these rows as shown in the sample outputs.

## Task A: One Box

| Input | Output |
| --- | --- |
|  |  |

## Task B: Three Boxes

| Input | Output |
| --- | --- |
|  |  |

## Code

```
Insert MATLAB code here.
```

```
Insert C/C++ code here.
```

# Part 3: Digit Segmentation

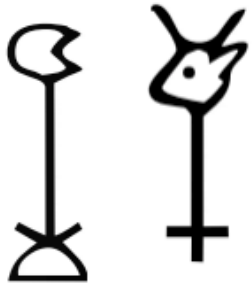
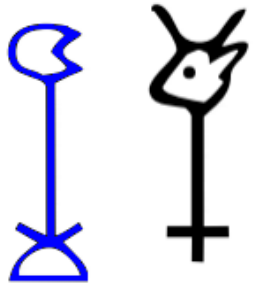
For Part 3 of this assignment, you will only need to produce one working program. It can be written in either MATLAB, C, or C++.

In each of these tasks, you will need to identify every instance of “known” hieroglyphs. Each of the four tasks expands this set of symbols. For each kind of hieroglyph, you should chose an arbitrary but distinct color, then fill all instances of this hieroglyph with that color.

You should develop and extend a single program to perform each subsequent task. Ideally, one would design it such that expanding the set of symbols will only require that you add additional templates.

## Task A: Lotus Flower

For this task, find and color the hieroglyph which represents the number one thousand. Do not color the other hieroglyph.

| Input | Output |
| --- | --- |
|  |  |

## Task B: Heel and Rope

For this task, you will need to distinctly color all instances of three hieroglyphs: ten, one hundred, and one thousand. Do not color any other hieroglyphs.

| Input | Output |
| --- | --- |
|  |  |

## Task C: Bones

This section introduces the values one through nine. Color each of the clusters with its own color.

| Input | Output |
| --- | --- |
|  |  |

## Task D: The Complete Set

This section introduces the rest of the numeric hieroglyphs: ten thousand, one hundred thousand, one million. After solving this task, your program should be able to find and distinctly color every numeric hieroglyph from one to one million.

| Input | Output |
| --- | --- |
|  |  |

## Code

```
Insert MATLAB code here.
```

```
Insert C/C++ code here.
```

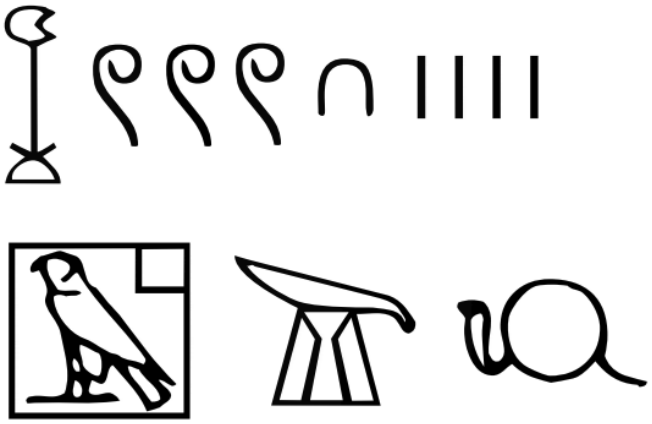
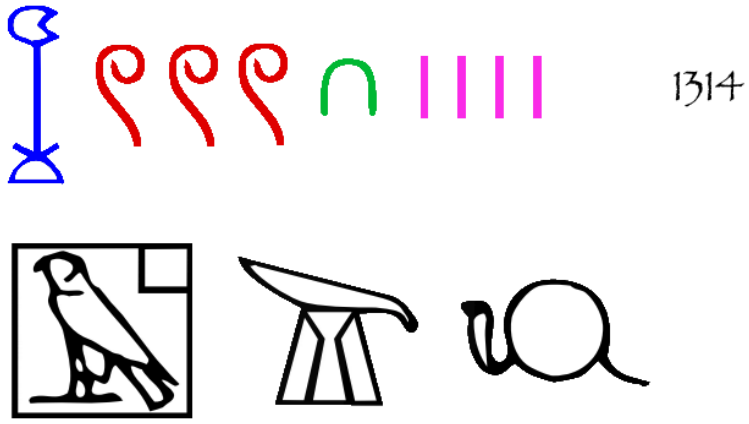
# Part 4: Number Segmentation

For Part 4 of this assignment, you will only need to produce one working program. It can be written in either MATLAB, C, or C++.

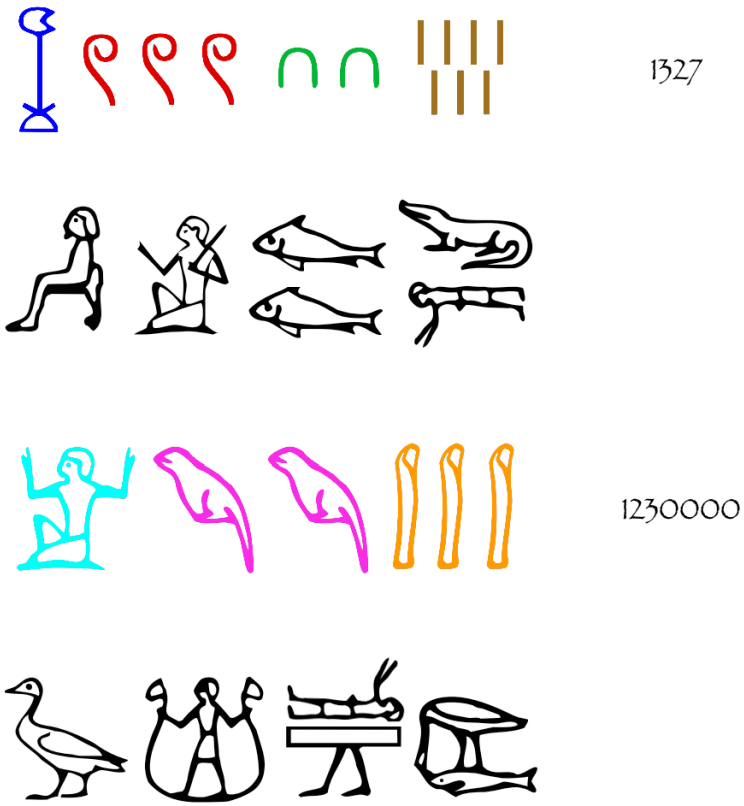
Notice that each row of hieroglyphs is either a sequence of non-numeral hieroglyphs or a valid number. For each row of numeral hieroglyphs, your program should parse the value represented and print it (with base ten arabic numerals) in the right margin. The rows of non-numeral hieroglyphs should be ignored.

For an explaination of how to interpret the ancient egyptian numbers, please see the resources listed at the top of this page.

## Task A

| Input | Output |
| --- | --- |
|  |   |

## Task B

| Input | Output |
| --- | --- |
|  |   |

## Code

```
Insert MATLAB code here.
```

```
Insert C++ code here.
```

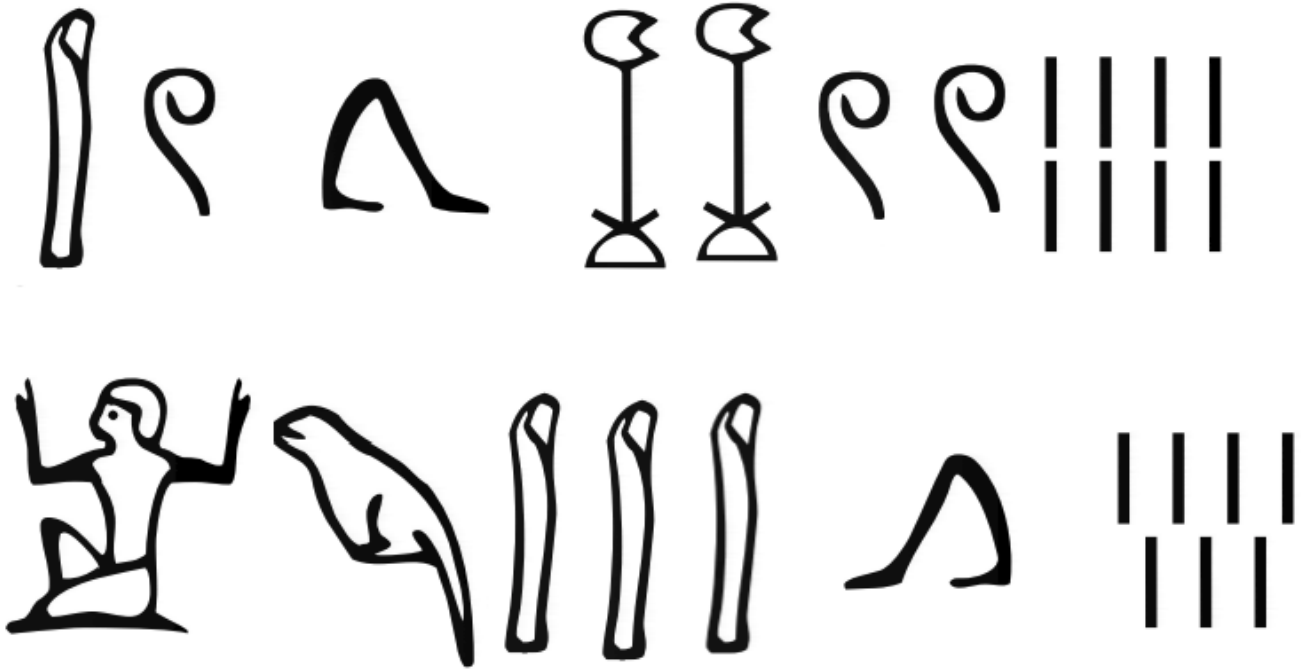
# Part 5: Basic Arithemetic (Extra Credit)

For Part 5 of this assignment, you will only need to produce one working program. It can be written in either MATLAB, C, or C++.

Notice that we have added two additional symbos: addition and subtraction. For each of the given images, parse each row and perform the described arithmetic operations. Write the computed values in the right-hand margin as shown.

## Task A: Two Rows, Two Operations

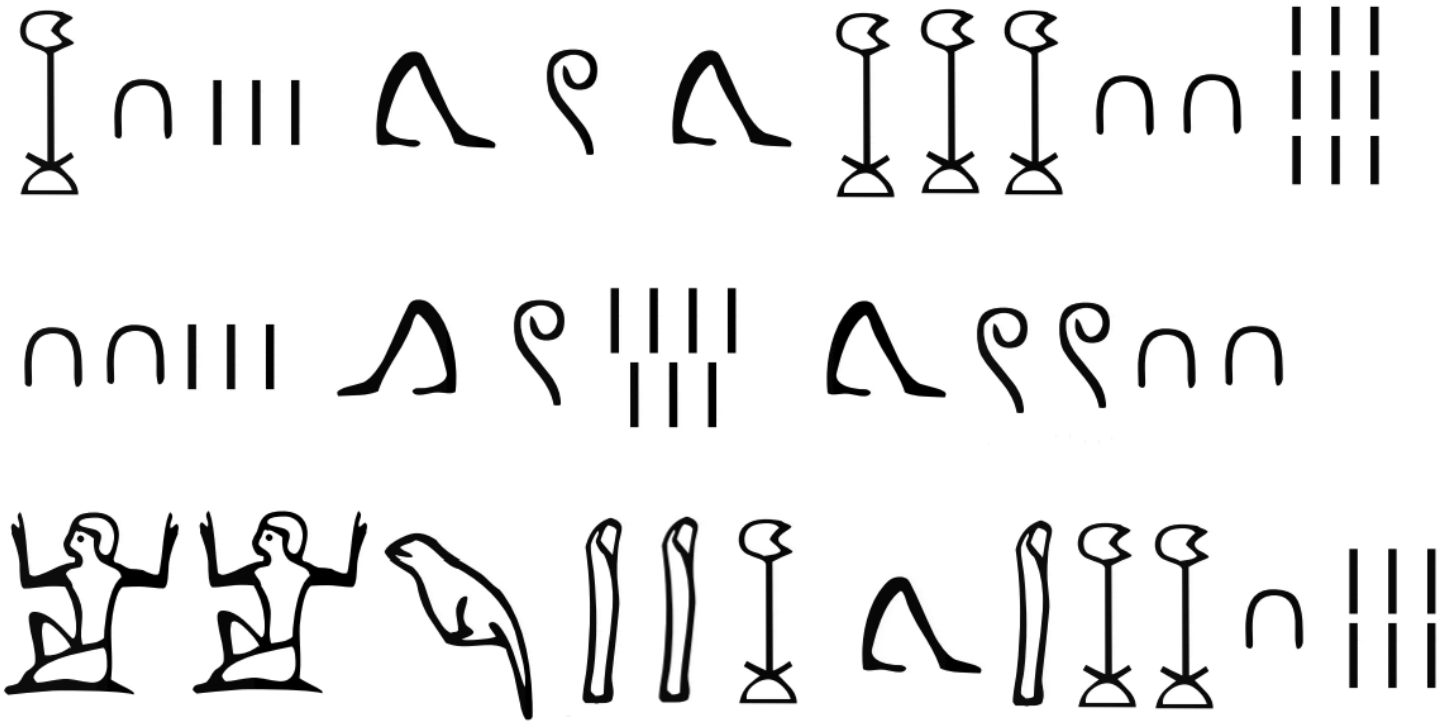
For the first row, add the LHS to the RHS; for the second row, subtract the RHS from the LHS.

| Input |
| --- |
|  |

# Task B: Arithmetic Parsing

For each row of the input file, parse and perform the described series of arithmetic operations. There may be an arbitrary number of rows, and on each row there may be an arbitrary number of arithmetic operations.

**Input**



# Code

```
Insert MATLAB code here.
```

```
Insert C++ code here.
```

# 6. Useful Links

Using OpenCV with gcc and CMake [http://docs.opencv.org/trunk/doc/tutorials/introduction/linux_gcc_cmake/linux_gcc_cmake.html]

Install OpenCV On OS X [http://www.vrac.iastate.edu/575x/S2012/doku.php?id=opencv_macports&DokuWiki=52ad1bc0d5c4d2225f87f47cf1612e88]

Using OpenCV with Visual Studio - Official OpenCV Wiki. [http://opencv.willowgarage.com/wiki/VisualC%2B%2B]

Useful openCV tutorials and code samples [http://www.vrac.iastate.edu/575x/S2012/doku.php?id=opencv_resources]

Inverting an image using OpenCV [http://opencv.wikispaces.com/4%29+Inverting+the+Image]

Finding connected components with OpenCV [http://superkkt.com/488]

Finding connected components with MATLAB [http://www.mathworks.com/help/toolbox/images/ref/bwconncomp.html]