***Written Assignment:***

*Description, Correctness, Time analysis, and Seam-carved Im- age: Write down a description of your algorithm. Add a paragraph prov- ing why your algorithm produces the right answer, and also provide a derivation of the running time in terms of the dimensions of the input image H (height) and W (width).*

* **Description: There are two algorithms that I developed, the first one is the energy algorithm and the second one is the vertical seam algorithm**
  + **Energy Algorithm:**
    - I create an empty numpy array to store the energy values of each pixel
    - We jump into a nested for loop (first iteration is height, second iteration is width). Within the for loop, I have an if else element.
      * If the pixel is any of the edges, then make the energy value equals to 1000
      * Else, calculate the energy using the equation given in the project description.
  + **Find Vertical Seam:**
    - I create an empty numpy array that can store the energy path values of all of the pixels and also the j value of each element of that exact energy pixel.
    - Again, we jump into a nested for loop with the first elements being the height and the second elements being the width. Within the second for loop, I have one if else element and an elif.
      * If(i == 0)
        + Seam = energy
        + Keeping the first energy value of every pixel and building up from it
      * elif(j ==(width - 1))/elif(j == 0)
        + seam = energy + min(energy value of neighbors with the exception of the last column since they won’t have a neighbor to the right or left)
      * else:
        + seam = energy + min(energy value of its neighbors)
    - After we’ve calculated the energy seam of all of the pixels, we find the index value j of the pixel with the least value in the seam array, and we iterate backwards in the seam array trying to find the path that leads to this min value. We store the jth values of each pixel in a one dimensional numpy array and that is what is returned by the vertical seam function
* **Correctness:**
  + The original goal of this algorithm was to build a program that could read and convert the RGB values of a picture into an energy image and then find vertical seams with the least amount of energy and then take them out. Through dynamic programming, the algorithm provided works, as it takes a numpy array of the original image, converts that into a numpy array of minimum energies, and finds the min seam through backtracking and finding the path taken to arrive at that minimum energy. The algorithm is also able to calculate this path if a horizontal seam is required.
* **Time analysis: There are three algorithms I created in thai project**
  + **Energy Calculator:**
    - Since we are calculating each pixel and we have a nested for loop to do so, our time complexity will be like so since we have a H\*W image:
      * Nested for loop: O(H\*W)
  + **Find Vertical Seam:**
    - Nested for loop for seam finding for each pixel: O(H\*W)
    - For loop for finding the j axis of the min value in the last row: O(W)
    - At the end, in addition to calculating the lowest seam energy, we also have to go up the entire height of the image and reconstruct the seam. Backtracking for loop: O(H)
    - Final runtime would be O(H\*W)
  + **whereHelper:** The whereHelper is what I used to find the min J axis of the chosen neighbors. It only compares three values (the 3 possible values/neighbors) and thus runs in O(3)