2. I implemented this using the filters package of scipy like the example code in the textbook on page 33.

3. This function was implemented in the following manner. For both vertical and horizontal seams we need to iterate over each pixel of the image. In the vertical case we start from the top left of the image and iterate left to right and then move to the next row. We set each pixel equal to the image magnitude’s value at that pixel plus the min of the upper left, upper middle, or upper right in our new cumulative energy map (CEM). If the upper left, middle, or right are out of the image’s bounds they evaluate to 0. In the horizontal case we also iterate starting from the top left, but this time moving down the column until the bottom of the image, and then on to the next column. We also set each pixel equal to the image magnitude’s value at that pixel, but this time plus the min of the upper left, left, and lower left indexes in our new CEM. The rules for out of bound indexes are the same as in the vertical case.

4. We can find the optimal vertical seam by first greedily choosing the min value of the bottom row of the CEM. Starting from this point we can find the min path by just choosing the min upper left, right, or middle index until we reach the top row.

5. We can find the optimal horizontal seam by first greedily choosing the min value of the rightmost column of the CEM. Starting from this point we can find the min path by just choosing the min upper left, left, or lower left index until we reach the leftmost row.

6. To display the seam over the image we need two slightly different approaches for vertical and horizontal seams. For vertical, we can iterate over each row of an image and set the row index stored in our seam at the current row index equal to red, or whichever color we choose to represent the seam. For horizontal we can apply the same logic, but iterating over columns instead of rows.

7. The greedy vertical seam can be obtained by iterating over each row and storing the index with the min value in our seam. The greedy horizontal seam can be obtained by iterating over each column and storing the index with the min value in our seam.

8. To reduce the size of the image we first need to get the optimal seam for the direction we wish to shrink. To reduce the size horizontally we get the optimal vertical seam and iterate over each row of the image and remove the index specified by the seam. To reduce vertically we get the optimal horizontal seam and then transpose our image so we can repeat the process as before, and then transpose again to return it to the original orientation.

9. 10. Use a for loop to call reduceImage 100 times