CS116 – Activity 9

Suppose we have an image with a white background and irregularly shaped blobs of some contrasting color (a medical image and tumors?) and we want to know the size, measured in the number of pixels, of one of the blobs. We could define a rectangular region around a blob and count the non-white pixels in that region. The irregular shapes may make it difficult to define a rectangular region which contains just the blob we want. We need to try another way.

Most programming languages allow a function to use itself. This is called recursion. In order for the recursion to not go on forever, a recursive function must have an if statement that checks for a condition for the recursion to terminate so recursive methods are always of the form:

```
def functionName(parameters):
   if (base condition):
      # return a value or do an operation
   else:
      # use the function with new (smaller) parameter values
      # and use this result in some way to solve the original problem
```

Now let's look at how recursion can make the blob pixel counting easy. We start with a parameter for a pixel inside the blob and that pixel gets counted. We then look at the neighbors of that pixel and we want to repeat the process for each of the neighbors as long as they are still within the blob. A pixel is still within the blob if it doesn't fall outside of the picture and that pixel is non-white. To repeat the process we simply use blobCount again. Does it feel weird to you to have a function use itself? That's okay. Most people get the feeling initially. Take a look at the function code:

```
def blob(pic,x,y):
  # if the pixel is outside the picture then don't count it
  if x<0 or y<0 or x>getWidth(pic) or y>getHeight(pic):
    return 0
  # if the pixel is white(ish) then we don't count it
  elif distance(getColor(getPixel(pic,x,y)),white) < 10:
    return 0
  else:
    # mark the pixel as white so it doesn't get counted again
    setColor(getPixel(pic,x,y),white)
    # count the pixel plus the blob counts of all its neighbors
    return 1 + blob(pic,x-1,y) + blob(pic,x+1,y) + blob(pic,x,y-1) + blob(pic,x,y+1)
```

Now, you will try to write your own recursive function! Suppose we want a function which is given a string and returns true or false on whether the string is a palindrome. A string is a palindrome if the string is the same as its reverse. One way to look at this problem is to realize that a string is a palindrome if the first and last letters are the same and the middle with the first and last letters removed is a palindrome. (See how I am using recursion there?).

Complete the following recursive code and try it out.

```
def palindrome(word):
  length = len(word)
  if ___________________:
     return True
  else:
     # complete the recursive case here
```