CS116 – Recursion

Purpose: Recursion is when a method or function invokes itself. There are times when iteration is difficult or impractical and recursion can achieve the same effect of doing a task multiple times.

Skills: After completion of this module you should be able to

1. Write a small recursive method

Schema:

```
def recursiveFunction(parameters):
    if base case condition:
        return a value or do an operation
    else:
        use recursiveFunction(smaller parameter values)
```

Activity: With your group perform the following tasks and answer the questions. You will be reporting your answers back to the class in 20 minutes.

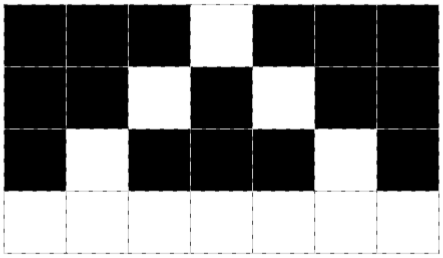
1. Suppose we have an image with a white background and irregularly shaped blobs of some contrasting color (a medical image and tumors?) and we want to know the size, measured in the number of pixels, of one of the blobs. Open up the picture `blobs.png` and examine the image.

   One possible approach is to define a rectangular region around a blob and count the non-white pixels in that region. What might make this approach difficult in counting the pixels in just one of the blobs?

   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 Picture method:

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

Create a Picture object with the blobs picture and try this method starting at pixel (400,20). To see what is happening we will consider a small image:



If we use the `blob` method at (5,1), what four recursive calls will be made?

What will happen with the recursive call at (4,1)? What is returned with this call?

What will happen with the recursive call at (5,2)? What is returned with this call?

The recursive call at (5,0) will make four recursive calls. What will happen for each of them and what will be returned for this call? Keep in mind that the previous recursive calls will set some pixels white so they won't be counted again.

The recursive call at (6,1) will make four recursive calls. What will happen for each of them and what will be returned for this call? Keep in mind that the previous recursive calls will set some pixels white so they won't be counted again.

Considering the results from the four recursive calls at (5,1), what is returned for this call?