Final Project (part 1) Script

```
"""
THIS SCRIPT TAKES IN A RASTER AND RETURNS TWO SHAPEFILES DIVIDING THE
CELLS OF THE RASTER TO CREATE POLYGONS
CENTERING AROUND LOCAL MINIMUM/MAXIMUM NODES.  THE FIRST SHAPEFILE IS
NONDISSOLVED
(AND SHOULD BE USED IN THE CORRESPONDING FINAL PROJECT PART 2 SCRIPT),
THE SECOND SHAPEFILE IS DISSOLVED.

To create an ArcToolbox tool with which to execute this script, do the following.
1   In  ArcMap > Catalog > Toolboxes > My Toolboxes, either select an existing toolbox
        or right-click on My Toolboxes and use New > Toolbox to create (then rename) a new
one.
2   Drag (or use ArcToolbox > Add Toolbox to add) this toolbox to ArcToolbox.
3   Right-click on the toolbox in ArcToolbox, and use Add > Script to open a dialog box.
4   In this Add Script dialog box, use Label to name the tool being created, and press Next.
5   In a new dialog box, browse to the .py file to be invoked by this tool, and press Next.
6   In the next dialog box, specify the following inputs (using dropdown menus wherever
possible)
        before pressing OK or Finish.
        DISPLAY NAME                    DATA TYPE    PROPERTY>DIRECTION>VALUE
        Input Raster                    Raster Layer   Input
        Output Shapefile NonDissolved   Shapefile            Output
        Output Shapefile Dissolved      Shapefile      Output


  To later revise any of this, right-click to the tool's name and select Properties.
"""

# Import necessary modules
import sys, os, string, math, arcpy, traceback, numpy
from arcpy import env
from arcpy.sa import *

# Check to see if Spatial Analyst license is available
if arcpy.CheckExtension("spatial") == "Available":

        # Allow output file to overwrite any existing file of the same name
        arcpy.env.overwriteOutput = True

        try:
        nameOfRaster                    = arcpy.GetParameterAsText(0)
```

```
	nameOfOutputShapefile		= arcpy.GetParameterAsText(1)
	nameOfOutputShapefileDis	= arcpy.GetParameterAsText(2)

	# Confirm import of raster file
	arcpy.AddMessage("Imported raster is named " + (nameOfRaster))
	arcpy.AddMessage("Output shapefile is named " + (nameOfOutputShapefile))

	#Start with input raster
	inputRaster = Raster(nameOfRaster)
	arcpy.AddMessage("inputRaster is named " + str(nameOfRaster))
	std = arcpy.GetRasterProperties_management(inputRaster, "STD").getOutput(0)
	arcpy.AddMessage("std of inputRaster is " + str(std))

	#Initialize polygonCount = numberOfCells in Raster
	columnCount = arcpy.GetRasterProperties_management(inputRaster,
"COLUMNCOUNT").getOutput(0)
	arcpy.AddMessage("columnCount is " + (columnCount))
	rowCount = arcpy.GetRasterProperties_management(inputRaster,
"ROWCOUNT").getOutput(0)
	arcpy.AddMessage("rowCount is " + (rowCount))
	cellCount = int(columnCount)*int(rowCount)
	arcpy.AddMessage("cellCount is " + str(cellCount))

	polygonCount = cellCount
	arcpy.AddMessage("polygonCount is " + str(polygonCount))

	#Initialize newRaster = a copy of inputRaster
	newRaster = inputRaster

	#Go through each cell in raster.

	maxFocalStat = FocalStatistics(inputRaster, NbrRectangle(3, 3, "CELL"), "MAXIMUM",
"DATA")
	minFocalStat = FocalStatistics(inputRaster, NbrRectangle(3, 3, "CELL"), "MINIMUM",
"DATA")
	meanMax = arcpy.GetRasterProperties_management(maxFocalStat,
"MEAN").getOutput(0)
	meanMin = arcpy.GetRasterProperties_management(minFocalStat,
"MEAN").getOutput(0)
	arcpy.AddMessage("first round min/max calculated")
	arcpy.AddMessage("mean of maxFocalStat is " + str(meanMax))
	arcpy.AddMessage("mean of minFocalStat is " + str(meanMin))
```

```
        #Determine if cell's value is highest (or equal to highest) or lowest (or equal to lowest)
among its queen contiguous neighbors.
        #If so, add it to localMinMax list, assign it a newRaster value of whatever its value was.

        newerRaster = Con(((maxFocalStat - inputRaster)>(inputRaster -
minFocalStat)),minFocalStat,maxFocalStat)
        newRaster = newerRaster

        arcpy.AddMessage("first round newRaster is calculated")
        std = arcpy.GetRasterProperties_management(newRaster, "STD").getOutput(0)
        arcpy.AddMessage("std of newRaster is " + str(std))

        newPoly = arcpy.RasterToPolygon_conversion(newRaster, nameOfOutputShapefile,
"NO_SIMPLIFY",
                        "VALUE")
        arcpy.AddMessage("new pre-dissolved shapefile is " + str(newPoly))
        numberOfRows = arcpy.GetCount_management(newPoly)
        arcpy.AddMessage("number of pre-dissolved features is" + str(numberOfRows))

        newPolyDis = arcpy.Dissolve_management(newPoly, nameOfOutputShapefileDis,
"gridcode", "", "",
                        "")
        arcpy.AddMessage("new post-dissolved shapefile is " + str(newPolyDis))
        numberOfRowsDis = arcpy.GetCount_management(newPolyDis)
        arcpy.AddMessage("number of post-dissolved features is" + str(numberOfRowsDis))

        counter = 0
        arcpy.AddMessage("start counter is" + str(counter))

        lastMaxFocalStat = 0

        while True:
        counter += 1
        arcpy.AddMessage("beginning iteration number" + str(counter))

        #Go through each cell in newRaster again.  Find highest and lowest value among queen
contiguous neighbors.
        #Assign to this cell the value of whichever of those two neighbors is closest to this cell's
previous value.

        maxFocalStat = FocalStatistics(newRaster, NbrRectangle(3, 3, "CELL"), "MAXIMUM",
"DATA")
```

```
        minFocalStat = FocalStatistics(newRaster, NbrRectangle(3, 3, "CELL"), "MINIMUM",
"DATA")
        meanMax = arcpy.GetRasterProperties_management(maxFocalStat,
"MEAN").getOutput(0)
        meanMin = arcpy.GetRasterProperties_management(minFocalStat,
"MEAN").getOutput(0)
        uniqueValCountMax = arcpy.GetRasterProperties_management(maxFocalStat,
"UNIQUEVALUECOUNT").getOutput(0)
        uniqueValCountMin = arcpy.GetRasterProperties_management(maxFocalStat,
"UNIQUEVALUECOUNT").getOutput(0)
        arcpy.AddMessage("mean of maxFocalStat is " + str(meanMax))
        arcpy.AddMessage("mean of minFocalStat is " + str(meanMin))
        arcpy.AddMessage("unique value count of maxFocalStat is " + str(uniqueValCountMax))
        arcpy.AddMessage("unique value count of minFocalStat is " + str(uniqueValCountMin))

        #Determine if cell's value is highest (or equal to highest) or lowest (or equal to lowest)
among its queen contiguous neighbors.
        #If so, add it to localMinMax list, assign it a newRaster value of whatever its value was.

        std = arcpy.GetRasterProperties_management(newRaster, "STD").getOutput(0)
        arcpy.AddMessage("std of newRaster is " + str(std))

        newerRaster = Con(((maxFocalStat - newRaster)>=(newRaster -
minFocalStat)),minFocalStat,maxFocalStat)
        newRaster = newerRaster

        std = arcpy.GetRasterProperties_management(newRaster, "STD").getOutput(0)
        arcpy.AddMessage("std of newRaster is " + str(std))

        newPoly = arcpy.RasterToPolygon_conversion(newRaster, nameOfOutputShapefile,
"NO_SIMPLIFY",
                                "VALUE")
        arcpy.AddMessage("new pre-dissolved shapefile is " + str(newPoly))
        numberOfRows = arcpy.GetCount_management(newPoly)
        arcpy.AddMessage("number of pre-dissolved features is" + str(numberOfRows))

        newPolyDis = arcpy.Dissolve_management(newPoly, nameOfOutputShapefileDis,
"gridcode", "", "",
                          "")
        arcpy.AddMessage("new post-dissolve shapefile is " + str(newPolyDis))
        numberOfRowsDis = arcpy.GetCount_management(newPolyDis)
        arcpy.AddMessage("number of features post-dissolve is" + str(numberOfRowsDis))
        arcpy.AddMessage("completed iteration number" + str(counter))

        #max allowable iteration counter included to protect against loops -- unlikely to hit this
counter.
        maxAllowableCounter = 1000

        if counter > maxAllowableCounter:
                arcpy.AddMessage("counter > " + str(maxAllowableCounter))
                break
        elif lastMaxFocalStat == meanMax:
                arcpy.AddMessage("lastMaxFocalStat = " + str() + " and current meanmax = " +
str(meanMax))
                break
        else:
                lastMaxFocalStat = meanMax
                continue

        arcpy.AddMessage("final counter is " + str(counter))
        arcpy.AddMessage("final number of rows is " + str(numberOfRowsDis))

        nameOfOutputShapefile = newPoly

        except Exception as e:
        # If unsuccessful, end gracefully by indicating why
        arcpy.AddError('\n' + "Script failed because: \t\t" + e.message )
        # ... and where
        exceptionreport = sys.exc_info()[2]
        fullermessage   = traceback.format_tb(exceptionreport)[0]
        arcpy.AddError("at this location: \n\n" + fullermessage + "\n")

else:
        # Report error message if Spatial Analyst license is unavailable
        arcpy.AddMessage ("Spatial Analyst license is unavailable")
```