GIS 540 Exam I

Course number: GIS 540

Test ID: exam I

Instructor: Dr. Tateosian

Time Limit: 120 minutes

Materials allowed: One hand written page of notes (front and back), scrap paper (students may not take note page or scrap paper away from the exam).

General instructions: **Write clearly**. Use complete sentences for short answer questions. Write name and unity login id (e.g. jkrowlin) on each page. Return all pages of the exam, note page, *and* scrap paper to the proctor. Do not discuss the exam with other students before the exams are returned. Write clearly and make sure that the difference between upper and lower case letters is easily distinguishable. Appendix I on the last page of the exam is provided as a reference.

Total points deducted ____________

Score ___________ /107 = ____________ %

Name___________________________________ NCSU Login ID _________________

1. **(20pts)** In the left table, match the Python expression with the term that describes it. In the right table, specify whether the code would evaluate as True or False if used as a Boolean expression in a conditional statement. There is one expression that can't be used this way; For this one, write ERROR. Assume that fileN is 'data1.shp', inF is 'data2.shp', and x is 5.

Match	**Python code**	**Description**
F	fileN[3]	A. assignment
G	fileN[:-4]	B. comparison
E	"workspace"	C. exponentiation
B	fileN == inF	D. integer division
A	fileN = inF	E. string literal
J	fileN + inF	F. indexing
I	x + 3	G. slicing
D	1/x	H. empty list
C	x**3	I. addition
H	[]	J. concatenation

Python code	**True or False?**
fileN[3]	T
fileN[:-4]	T
"workspace"	T
fileN == inF	F
fileN = inF	ERROR
fileN + inF	T
x + 3	T
1/x	F
x**3	T
[]	F

2. **(20pts)** Label each item in the list as **A** for arcpy method, **F** for built-in function, **K** for keyword, **L** for list method, **M** for built-in module, **O** for os.path method, or **S** for string method. *There are 2 built-in modules and three of each of the rest.*

__A_ Describe	_L__ reverse	__F_ float	__O_getmtime	__S__ strip
__M__ sys	_A__CheckOutExtension	_S__ replace	_K__ if	_O__ splitext
_O__ dirname	_K__ while	_L__ append	_S__ format	_L__ sort
_F__ int	_M__ os	_K__ or	_F__ range	_A__Buffer_analysis

3. **(3pts)** Write a Python WHILE-loop equivalent to the following FOR-loop.

FOR-loop	WHILE-loop

```
for x in range(2,12,2):
    print x
```
	```
	x = 2
	while x < 12:
	    print x
	    x = x + 2
	```

Name___________________________________ NCSU Login ID _________________

4. **(9pts)**Use the script arguments on the left and the code fragments in the middle in the table below to give the output. If the code would cause an error, write *ERROR* and give a *brief explanation*. (Assume that sys and os have already been imported.)

Arguments	**Python code**	**Output or cause of error**

8 5	a = sys.argv[1]
	b = sys.argv[2]
	print a + b
		85

C:/data p.shp 50	d = sys.argv[1]
	f = sys.argv[2]
	u = sys.argv[3]
	print '{0}/{1} has {2} polys'.format(d,f,u)
		C:/data/p.shp has 50 polys

8	myList = [7, 4, 6] + [int(sys.argv[1])]
	max = max(myList)
	print max
		ERROR; The built-in ' max' function was overwritten by setting it to an integer on the 2nd line.

35	latInput = int(sys.argv[1])
	latNC = 35.78
	if latInput < int(latNC):
	    print 'South'
	else:
	    print 'North'
		North (sys.argv[1] and latNC both become 35 when cast to integer, so the first condition is not true)

SUBDIVISION	fd = sys.argv[1]
	fd.lower( )
	print fd
		SUBDIVISION

a.txt b.shp c.qt	argList = sys.argv
	for f **in** argList:
	    print os.path.splitext(f)[1],
		.py .txt .shp .qt

A B C	myList = []
	myList.append(sys.argv[1])
	myList.append(sys.argv[2])
	myList.append(sys.argv[3])
	myList.reverse()
	print myList
		[' C' ,' B' ,' A']

C:/data/bird.JPG	dsc = arcpy.Describe(sys.argv[1])
	if dsc.dataType == "ShapeFile" **or** "Folder":
	    print "Sh or Fo"
	else:
	    print "Other"
		Sh or Fo (" Folder" is True, need to use full Boolean statement on both sides of the comparison operator. E.g., dsc.dataType == "ShapeFile" or dsc.dataType == "Folder")

GIS	**print** 'I'm good at {0}'.format(sys.argv[1])	ERROR ' I' interpreted as the end of the string literal, so the rest (m good at {0}) can't be interpreted.

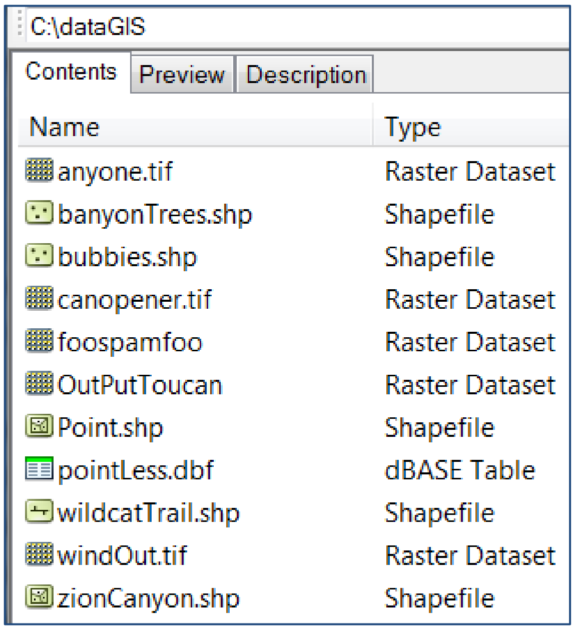
Name___________________________________ NCSU Login ID _________________

5. **(4pts)** Under each arcpy listing method in the table below, give the list contents. If the list is empty, write EMPTY and give a brief explanation. Assume the user passed in the following arguments:

   C:/dataGIS anyo *one

   Also assume the following code has been executed:

   import arcpy, sys
   arcpy.env.workspace = sys.argv[1]
   subStr = '*{0}*'.format(sys.argv[2])

L1 = arcpy.ListRasters("*", "GRID")	
foospamfoo OutPutToucan	
L2 = arcpy.ListTables("wind*")	
EMPTY; no table names start with wind	
L3 = arcpy.ListRasters("sys.argv[3]")	
EMPTY; No rasters are named "sys.argv[3]" Probably intended to remove the quotation marks.	
L4 = arcpy.ListFeatureClasses(subStr)	
banyonTrees.shp, zionCanyon.shp	

C:\dataGIS

Name	Type
anyone.tif	Raster Dataset
banyonTrees.shp	Shapefile
bubbies.shp	Shapefile
canopener.tif	Raster Dataset
foospamfoo	Raster Dataset
OutPutToucan	Raster Dataset
Point.shp	Shapefile
pointLess.dbf	dBASE Table
wildcatTrail.shp	Shapefile
windOut.tif	Raster Dataset
zionCanyon.shp	Shapefile

Figure 1: Contents of C:\dataGIS in ArcCatalog.

6. **(3pts)** Answer the following questions about the traceback error shown in the box below:
   a. Which line number of 'convertLat.py' did the error occur on? **23**

   b. What is the name of the exception? **NameError**

   c. Explain why the exception was thrown. **The variable, 'degree' is being used without being defined.**

```
Traceback (most recent call last):
  File "C:\Python27\ArcGIS10.2\Lib\site-packages\scriptutils.py", line 326, in RunScript
    exec codeObject in __main__.__dict__
  File "C:\temp\convertLat.py", line 23, in <module>
    decimalDegrees = float(degree + (minute/60 + second/3600))
NameError: name 'degree' is not defined
```

Name___________________________________ NCSU Login ID _________________

7. **(6pts)** Complete the script, addYears.py, started in the box below. It should take two user arguments: a date stamp (formatted YYYY-DD-MM) and an integer, x. The code should print a date stamp with x years added to the original date stamp. For example, if the user enters 2011-01-28 and 5, the script should print "The updated date stamp is 2016-01-28".

```
# addYears.py  (Alternative solution shown in green comments)

import sys

dateStamp = sys.argv[1]

x = int(sys.argv[2])
# year = dateStamp.split('-')[0]
year = dateStamp[:4]

year = int(year)

year = year + x
# newDateStamp = str(year) + '-' + dateStamp.split('-')[1] + '-' +dateStamp.split('-')[2]
newDateStamp = str(year) + dateStamp[4:]

print 'The updated date stamp is {0}'.format(newDateStamp)
```

8. **(9pts)** This script should take a polygon shapefile and set the values of its 'sqkmArea' field to the polygon areas (in km^2). If the field does not exist yet, the script should create the field. In the box below, correct the 9 mistakes in the code.

```
import arcpy, sys

inputFile = ~~sys.argv(1)~~ sys.argv[1]

~~areaField == 'sqkmArea'~~  areaField = 'sqkmArea'

~~fields = arcpy.ListFields()~~  fields = arcpy.ListFields(inputFile)

nameList = ~~[f.type for f in fields]~~  [f.name for f in fields]

~~if areaField in nameList:~~   if areaField not in nameList:

→arcpy.AddField_management~~(areaField, 'Double')~~   (inputFile, areaField, 'Double')

expression = ~~"!shape.length@squarekilometers!"~~ "!shape.area@squarekilometers!"

arcpy.CalculateField_management(inputFile, areaField, expression, 'PYTHON')
~~print "Field {0} calculated in the input shapefile {1}".format(inputFile, areaField)~~
print "Field {0} calculated in the input shapefile {1}".format(areaField, inputFile)
```

Name__________________________________ NCSU Login ID _________________

1. **(7pts)** The script ‘PointsToLine_nonBatch.py’ in the first box below creates a Shapefile with lines connecting the points in the ‘C:/data/nests.shp’ file. Rewrite this as ‘PointsToLine_Batch.py’ in the second box. The batch version should call the Points to Line (Management) tool on all the Point Shapefiles in the ‘C:/data’ directory whose names contain the substring “tucan”.

```
# PointsToLine_nonBatch.py
import arcpy, os
arcpy.env.workspace = 'C:/data'
inputFile = 'nests.shp'
outputFile = os.path.splitext(inputFile)[0] + 'Line.shp'
arcpy.PointsToLine_management(inputFile, outputFile)
print '{0} created.'.format(outputFile)
```

```
# PointsToLine_Batch.py
import arcpy, os
arcpy.env.workspace = 'C:/data'
fcs = arcpy.ListFeatureClasses('*tucan*', 'Point')
for f in fcs:
    outputFile = os.path.splitext(f)[0] + 'Line.shp'
    arcpy.PointsToLine_management(f, outputFile)
    print '{0} created.'.format(outputFile)
```

Name___________________________________ NCSU Login ID _________________

10. **(8 pts)** Fill in the blanks to complete getSize.py. Use sys.argv, os.path.dirname, os.listdir, and os.path.getsize to print the size of each of the files in the directory where the script resides.

```
# getSize.py

import  os, sys

scriptPath = sys.argv[0]

scriptDirectory =  os.path.dirname(scriptPath)

fileList = os.listdir(scriptDirectory)

for f in fileList:

    fullPathFileName = scriptDirectory + '/' + f

    theSize = os.path.getsize(fullPathFileName)

    print 'The size of {0} is {1}'.format(f, theSize)
```

Name__________________________________ NCSU Login ID _________________

10. **(18pts)** By design, handleShapes.py should determine the data type of an input file. If the input file is a polygon shapefile with more than 50 records, the script should find a point inside each polygon. If it's a point shapefile, the script should find the mean center of the points. Answer questions a)-g) pertaining to handleShapes.py.

```
1   # handleShapes.py
2   import arcpy, sys
3   inputFile = sys.argv[1]
4   dsc = arcpy.~~env.~~Describe(inputFile)
5   count = arcpy.GetCount_management(inputFile)
6   outputFile = inputFile[:-4] + 'out.shp'  #CODE MOVED OUTSIDE THE CONDITIONAL BRANCH
7   if (dsc.dataType == 'ShapeFile' and dsc.shapeType == 'Polygon') and count > 50:
8       arcpy.FeatureToPoint_management (inputFile, outputFile, 'INSIDE')
9   elif dsc.dataType == 'ShapeFile' and dsc.shapeType == 'Point':
10      arcpy.MeanCenter_stats(inputFile, outputFile)
11  →print "Mean center calculated for {0}".format(inputFile)
```

a) The Get Count tool is in the ____Data Management__toolbox and the Mean Center tool belongs to the ___Spatial Statistics_____toolbox.

b) Which (if any) ArcGIS environment settings does handleShapes.py set?
None

c) Line 6 has a logic error involving a compound conditional keyword. Describe the characteristics files for which this statement is true.
--Any dataset with more than 50 records.
--All polygon shapefiles, including those with less than 50 records.

d) If the value of inputFile is 'C:/data/park.shp' and outputFile is set as shown on line 7, what is the value of outputFile?
C:/data/park.shpout.shp

e) If outputFile is set on line 7, what is the value of outputFile when the script reaches line 10?
It would not be defined.

f) If the rest of the script works, under what conditions would line 11 report that the mean center has been calculated?
For any input that didn't cause an error to be thrown, this statement would be printed.

g) Inside the box, show how to correct the 10 errors in this script.

Appendix 1 arcpy and os function signatures.

AddField_management (in_table, field_name, field_type, {field_precision}, {field_scale}, {field_length}, {field_alias}, {field_is_nullable}, {field_is_required}, {field_domain})

CalculateField_management (in_table, field, expression, {expression_type}, {code_block})

MeanCenter_stats (Input_Feature_Class, Output_Feature_Class, {Weight_Field}, {Case_Field}, {Dimension_Field})

GetCount_management (in_table)

PointsToLine_management (Input_Features, Output_Feature_Class, {Line_Field}, {Sort_Field}, {Close_Line})

ListFeatureClasses ({wild_card}, {feature_type}, {feature_dataset})

ListFields (dataset, {wild_card}, {field_type})

ListRasters ({wild_card}, {raster_type})

ListTables ({wild_card}, {table_type})

os.listdir(path)

os.path.dirname(path)

os.path.getsize(path)

os.path.splitext(path)