```
# ***************************************************************************
#
# NAME: Your Name
# DATE: date
# CLASS: GEOG410
# ASSIGNMENT: Lab #
#
# DESCRIPTION: describe the script, what it does, how it does it,
#       and any other important information
#
# INSTRUCTIONS: usage instructions, i.e., inputs, outputs, and how to
#       run the script
#
# ***************************************************************************

# ********** IMPORT STATEMENTS **********
import sys
import os
import argparse
import arcpy
from arcpy import env


# ********** GLOBAL CONSTANTS **********

POINTS = "points"
CITIES = "cities"
REQUIRED = [POINTS, CITIES]

OUTPUT = "suitable"

ACCEPTABLE_VALUES = (38, 17)
MAX_DIST = 4000

SPATIAL_ANALYST = "Spatial"


# ********** FUNCTIONS **********

def parse_args(argv):
    """
    """
    a = argparse.ArgumentParser()
    a.add_argument("geodatabase", type=valid_geodatabase)
    b = a.parse_args(argv)
    return b.geodatabase


def valid_geodatabase(path):
    """
    """
    for dataset in REQUIRED:
        if not arcpy.Exists(os.path.join(path, dataset)):
            raise argparse.ArgumentTypeError("Geodatabase provided is not valid.")
    return path


def function_1(arg1, template_raster=None):
    """
    """
    #
    b = arcpy.sa.EucDistance(arg1, cell_size=template_raster)
    return b
```

```
def function_2(arg1, arg2):
    """
    """
    #
    a = arcpy.sa.Con(arg1, 0, 0)
    #
    for i in arg2:
        #
        a |= arg1 == i
    return a


# ********** MAIN **********

def main(argv):
    #
    env.overwriteOutput = True

    #
    z = parse_args(argv)

    #
    points = os.path.join(z, POINTS)
    cities = os.path.join(z, CITIES)
    output = os.path.join(z, OUTPUT)

    #
    cities_raster = arcpy.Raster(cities)

    #
    if arcpy.CheckExtension(SPATIAL_ANALYST) == "Available":
        #
        arcpy.CheckOutExtension(SPATIAL_ANALYST)
    else:
        #
        raise LicenseError("Spatial Analyst Extension not available.")

    #
    try:
        #
        env.extent = cities_raster.extent
        #
        e = function_1(points, cities_raster)
        #
        f = function_2(cities_raster, ACCEPTABLE_VALUES)
        #
        g = (e <= MAX_DIST) & f
        #
        g = arcpy.sa.SetNull(g, 1, "VALUE = 0")
        #
        g.save(output)
    #
    finally:
        #
        arcpy.CheckInExtension(SPATIAL_ANALYST)

    return 0


# ********** MAIN CHECK **********

if __name__ == '__main__':
    sys.exit(main(sys.argv[1:]))
```