In [1]:
```
import requests
import zipfile
import os

# Function to download and extract a dataset
def download_and_extract(url, target_path, extract_path):
    response = requests.get(url)
    if response.status_code == 200:
        with open(target_path, 'wb') as file:
            file.write(response.content)
        with zipfile.ZipFile(target_path, 'r') as zip_ref:
            zip_ref.extractall(extract_path)

# URLs for the datasets
landcover_url = 'https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_dnr/bi
elevation_url = 'https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_dnr/el
counties_url = 'https://resources.gisdata.mn.gov/pub/gdrs/data/pub/us_mn_state_dnr/bdr

# Directory where you want to save the datasets
base_dir = r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2"

# Create new directories for unzipped data
unzipped_landcover_dir = os.path.join(base_dir, 'Unzipped Landcover')
unzipped_elevation_dir = os.path.join(base_dir, 'Unzipped Elevation')
unzipped_counties_dir = os.path.join(base_dir, 'Unzipped Counties')

# Ensure the new directories exist
os.makedirs(unzipped_landcover_dir, exist_ok=True)
os.makedirs(unzipped_elevation_dir, exist_ok=True)
os.makedirs(unzipped_counties_dir, exist_ok=True)

# Download and extract the landcover dataset
landcover_path = os.path.join(base_dir, 'landcover.zip')
download_and_extract(landcover_url, landcover_path, unzipped_landcover_dir)

# Download and extract the elevation dataset
elevation_path = os.path.join(base_dir, 'elevation.zip')
download_and_extract(elevation_url, elevation_path, unzipped_elevation_dir)

# Download and extract the counties dataset
counties_path = os.path.join(base_dir, 'counties.zip')
download_and_extract(counties_url, counties_path, unzipped_counties_dir)

print("Landcover, elevation, and counties datasets have been downloaded and extracted
```

```
Landcover, elevation, and counties datasets have been downloaded and extracted into s
eparate folders.
```

In [2]:
```
import os
import shutil

# Directory where the "Unzipped Landcover," "Unzipped Elevation," and "Unzipped Counti
base_dir = r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2"

# Define the new folder name
merged_folder_name = 'Merged Data'  # Change this to the desired name
```

```
# Create a directory for the merged data
merged_folder = os.path.join(base_dir, merged_folder_name)
os.makedirs(merged_folder, exist_ok=True)

# List the subdirectories to merge
subdirectories = ['Unzipped Landcover', 'Unzipped Elevation', 'Unzipped Counties']

# Iterate through the subdirectories and copy their contents to the merged folder
for subdirectory in subdirectories:
    subdirectory_path = os.path.join(base_dir, subdirectory)
    if os.path.exists(subdirectory_path):
        for root, dirs, files in os.walk(subdirectory_path):
            for file in files:
                file_path = os.path.join(root, file)
                shutil.copy(file_path, merged_folder)

print(f"Data from 'Unzipped Landcover,' 'Unzipped Elevation,' and 'Unzipped Counties'
```

```
Data from 'Unzipped Landcover,' 'Unzipped Elevation,' and 'Unzipped Counties' has bee
n merged into 'Merged Data'.
```

In [17]:

```
import arcpy

# Set the workspace and input feature class
arcpy.env.workspace = r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\La
input_feature_class = "mn_county_boundaries"

# Define the SQL expression to select features with CTY_Name values of 'Wabasha' or 'W
sql_expression = "CTY_Name IN ('Wabasha', 'Winona', 'Olmsted')"

# Create a feature layer with the selection
arcpy.management.MakeFeatureLayer(input_feature_class, "Selected_Counties", where_clau

# Specify the output feature class for the selected features
output_feature_class = r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\L

# Use the Clip tool to clip the selected features to themselves
arcpy.analysis.Clip(
    in_features="Selected_Counties",
    clip_features="Selected_Counties",
    out_feature_class=output_feature_class,
    cluster_tolerance=None
)
# Clear the selection
arcpy.management.SelectLayerByAttribute("Selected_Counties", "CLEAR_SELECTION")

print("Clipping Wabasha and Winona and Olmsted counties completed.")
```

```
Clipping Wabasha and Winona and Olmsted counties completed.
```

In [18]:

```
import arcpy
# Set the workspace and input feature class
arcpy.env.workspace = r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\La
input_feature_class = "mn_county_boundaries"

# Define the SQL expression to select features with CTY_Name values of 'Wabasha' or 'W
sql_expression = "CTY_Name IN ('Wabasha', 'Winona', 'Olmsted')"
```

```
# Create a feature layer with the selection
arcpy.management.MakeFeatureLayer(input_feature_class, "Selected_Counties", where_clau

# Specify the output feature class for the selected features
output_feature_class = r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\L

# Clear the selection
arcpy.management.SelectLayerByAttribute("Selected_Counties", "CLEAR_SELECTION")

print("Clipping and selecting Wabasha and Winona counties completed.")
```

```
Clipping and selecting Wabasha and Winona counties completed.
```

In [ ]:
```
#Clipped it out to counties
out_raster = arcpy.sa.ExtractByMask(
    in_raster="NLCD_2019_Land_Cover.tif",
    in_mask_data="Selected_Counties",
    extraction_area="INSIDE",
    analysis_extent='189775.332 4816305.37 761655.0734 5472427.737 PROJCS["NAD_1983_UT
)
out_raster.save(r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\Lab_Part
```

In [ ]:
```
#Clipped it out to counties
out_raster = arcpy.sa.ExtractByMask(
    in_raster="digital_elevation_model_30m",
    in_mask_data="Selected_Counties",
    extraction_area="INSIDE",
    analysis_extent='189775.332039 4816305.370038 761655.0734 5472427.737 PROJCS["NAD_
)
out_raster.save(r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\Lab_Part
```

In [8]:
```
#identifies Slopes
out_raster = arcpy.sa.Slope(
    in_raster="Extract_digi2",
    output_measurement="DEGREE",
    z_factor=1,
    method="PLANAR",
    z_unit="METER",
    analysis_target_device="GPU_THEN_CPU"
)
out_raster.save(r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\Lab_Part
```

Out[8]:

Messages

In [ ]:
```
#Created Piont features for Dorys Home and North Picnic Area
arcpy.management.CreateFeatureclass(
    out_path=r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\Lab_Part2
    out_name="Dory",
    geometry_type="POINT",
    template=None,
    has_m="DISABLED",
    has_z="DISABLED",
    spatial_reference='PROJCS["NAD_1983_UTM_Zone_15N",GEOGCS["GCS_North_American_198
    config_keyword="",
```

```
    spatial_grid_1=0,
    spatial_grid_2=0,
    spatial_grid_3=0,
    out_alias=""
)
```

In [20]:
```
#reclassified slope
out_raster = arcpy.sa.Reclassify(
    in_raster="Slope_Extrac2",
    reclass_field="VALUE",
    remap="0 3.113066 1;3.113066 7.160053 2;7.160053 13.074879 3;13.074879 26.461065
    missing_values="DATA"
)
out_raster.save(r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\Lab_Pa
```

In [ ]:
```
#reclassified Landcover
out_raster = arcpy.sa.Reclassify(
    in_raster="Extract_NLCD3",
    reclass_field="NLCD_Land",
    remap="'Open Water' 5;'Developed, Open Space' 1;'Developed, Low Intensity' 1;'De
    missing_values="DATA"
)
out_raster.save(r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\Lab_Pa
```

In [5]:
```
#changes weights
import arcpy
import os

# Set the workspace where your rasters are located
arcpy.env.workspace = r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\
output_folder = r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\Lab_Pa

# List of input raster names
input_raster_names = ["Reclass_Slop6", "Reclass_Extr3"]

# Define weight scenarios
weight_scenarios = [0.25, 0.5]

# Nested loop to process each combination
for raster1_name in input_raster_names:
    for raster2_name in input_raster_names:
        for weight in weight_scenarios:
            if weight == 0.5:
                output_name = f"LandUse_Slope_EqualWeight"
            else:
                output_name = f"{raster1_name}_w{int(weight * 100)}_{raster2_name}_w
            #Skip if loop wants to pair the same rasters together
            if raster1_name == raster2_name:
                continue

            # Paths to Rasters
            raster1 = os.path.join(arcpy.env.workspace, raster1_name)
            raster2 = os.path.join(arcpy.env.workspace, raster2_name)

            # Create raster combinations
            raster1_weighted = arcpy.Raster(raster1) * weight
```

```
            raster2_weighted = arcpy.Raster(raster2) * (1 - weight)
            output_raster = raster1_weighted + raster2_weighted

            # Save the output raster
            output_raster.save(os.path.join(output_folder, output_name))
```

In [ ]:
```
#Path For Equal Weight Scenario
arcpy.sa.CostConnectivity(
    in_regions="Dory",
    in_cost_raster="LandUse_Slope_EqualWeight",
    out_feature_class=r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\
    out_neighbor_paths=None
)
```

In [ ]:
```
#Path for Slope Weight Scenario
arcpy.sa.CostConnectivity(
    in_regions="Dory",
    in_cost_raster="Reclass_Extr3_w25_Reclass_Slop6_w75",
    out_feature_class=r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\
    out_neighbor_paths=None
)
```

In [ ]:
```
#Path for Landcover Weight Scenario
arcpy.sa.CostConnectivity(
    in_regions="Dory",
    in_cost_raster="Reclass_Slop6_w25_Reclass_Extr3_w75",
    out_feature_class=r"C:\Users\Track\OneDrive\Documents\ArcGIS\Projects\Lab_Part2\
    out_neighbor_paths=None
)
```