For this practical, you will be designing one soft constraint. It is highly encouraged to leverage algorithms developed last week.

**Data Set Description**

I have provided two data sets, TestSet is smaller and it is recommend during initial design. BrainParcellation should be used for evaluating runtime in Task 3 and 4. FakeBrain.nii.gz is the original image, and each other image is a binary mask, the name corresponds to the structure. Target Points (targets.fcsv) and Entry points (entries.fcsv) are [MarkupFiducials](https://apidocs.slicer.org/v4.8/classvtkMRMLMarkupsNode.html#a98e21fa54198551a3884267d83e3c24a) listing possible entry and target points, you can pair these to create possible trajectories to consider.

For distance computation SimpleITK has several nice filters. As shown on the Course GitLab account you can leverage these filters using the following code:

class ComputeDistanceImageFromLabelMap():

def Execute(self, inputVolume):

sitkInput = su.PullVolumeFromSlicer(inputVolume)

distanceFilter = sitk.DanielssonDistanceMapImageFilter()

# you can try other filters from here: https://itk.org/Doxygen/html/group\_\_ITKDistanceMap.html

sitkOutput = distanceFilter.Execute(sitkInput)

outputVolume = su.PushVolumeToSlicer(sitkOutput, None, 'distanceMap')

return outputVolume

**Task Description**

Task 1a. Design an algorithm to find the closest blood vessel to a given path.

Start by writing the steps in PseudoCode (the use of a flow chart is encouraged). Use this to identify classes/functions to properly divide the code

Task 1b. Design a test to verify Task 1a is correct. You can create your own test data or use test data provided as part of the class exercise. How do you know the test returns the correct result?

Task 2. Find the best trajectory (entry-target fiducial combination) using a brute force approach. Return the corresponding points and distance value. Consider functions developed in Task 1 as helper functions for this routine.

Task 3a. Design another method to optimise entry-target point selection. One thing to consider is entry points can be grouped into clusters: to reduce potential search space.

Task 3b. How can the optimisation routine be validated (remember the code you created in Task 2….)

Note for the first short report Task 2 is sufficient to compute the result.