3rd party libraries

1. (10p) k-means clustering with dlib.
   Read **points.txt** into a container. It contains points in two clusters as shown in **clustering.jpg**. Use the k-means clustering algorithm of dlib to find the two centers of the clusters. Print the centers.
   See the slides about how to install 3rd party libraries and dlib. You will probably need dlib's headers "matrix.h" and "clustering.h". It may be helpful to treat individual data points as objects of type dlib::matrix<double,2,1> and store them into an std::vector. Similarly, you can prepare an std::vector of cluster centers which are also of the type dlib::matrix<double,2,1>. You can then find the centers using the dlib::find_clusters_using_kmeans() function.
   Helpful documentation and examples:
   http://dlib.net/ml.html#find_clusters_using_kmeans
   http://dlib.net/dlib/test/kmeans.cpp.html
2. (15p) Neural network classification with dlib.
   Read **points.txt** into a container. The points in the first 50 rows belong to one cluster and the points in the remaining 50 rows belong to another cluster.
   Use samples in rows 1-40 and 51-90 to train the network so that the points in rows 1-40 are trained to label (output value) 0 and the points in rows 51-90 are trained to label 1.
   Then test your network by predicting and printing the label for the points in rows 41-50 and 91-100. You should get a prediction close to 0 for the points in rows 41-50 and a prediction close to 1 for the points in rows 91-100. Print the predictions.
   You will probably need dlib's headers "matrix.h" and "mlp.h". It may be helpful to treat individual data points as objects of type dlib::matrix<double,2,1> and store them into an std::vector. You can create a multiplayer perceptron using the dlib::mlp::kernel_1a_c object. You can use its train() function to teach the network the labels associated with different data points. You can predict the label by calling the name of the mlp object.
   When creating the multilayer perceptron object for this problem, you should use a very small number of nodes and just one hidden layer.
   Helpful documentation and examples:
   http://dlib.net/ml.html#mlp
   http://dlib.net/mlp_ex.cpp.html