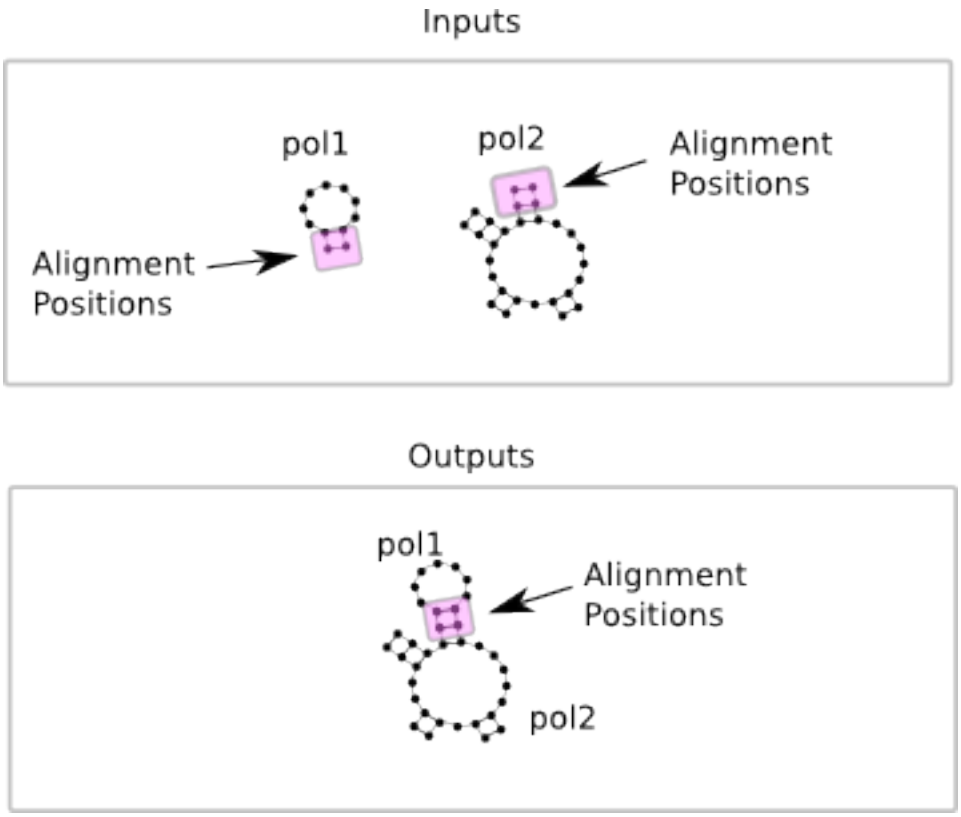
Challenge #1: Structure alignments
In this challenge you will write a function for the Polymere class that takes as inputs another Polymere class member and a vector of pairs of residue positions. Residue positions are represented by circles in the figure below. The program will align the two Polymere class members such that the root mean square distance between the pairs of residues is minimized. The coordinates of both Polymere members are updated to reflect the alignment. The image below shows a 2D representation of the problem.



Detailed Description
If you haven't already downloaded the RSIM source code use the following to create a new directory for the challenge project.

> git clone http://www.github.com/jpbida/RSIM

You will edit two files

RSIM/src/polymere.cpp
RSIM/apps/align.cpp

To solve this challenge you will add code to the Polymere::alignHelix function in the polymere.cpp file.

Programming Workflow
Your general workflow should be as follows.

1) Edit the polymere.cpp class alignHelix function.
2) build the project with the command

>cd RSIM/apps
>make align

3) run the align application using the test file.

>cd RSIM/tests
>./align.sh

4) Check to see if the output is correct.

pymol converted1.pdb

then load converted2.pdb and you should see two trna molecules joined together at a helix.

The polymere class stores the coordinates of atoms for the structure of an RNA molecule. These coordinates are loaded from PDB files that are generated from experimental approaches or simulations(www.pdb.org) The polymere class stores a low resolution representation that contains enough information to generate a high resolution(all atom) representation. The low resolution representation is stored as an array of of the Molecule class.

Low resolution points can be accessed with the following:
pol->mols[i]->x
pol->mols[i]->y
pol->mols[i]->z
pol->mols[i]->bx
pol->mols[i]->by
pol->mols[i]->bz
pol->mols[i]->b2x
pol->mols[i]->b2y
pol->mols[i]->b2z

The high resolution information is stored as a vector of doubles in

pol->full_atoms[i]

where

x=full_atoms[i]
y=full_atoms[i+1]
z=full_atoms[i+2]

For each low resolution mols[i] there are 27*3 entries in full_atoms.

Some useful functions that already exist in the Polymere class are Polymere::rmsd and Polymere::updateFull().

Polymere::rsmd
The function rmsd in the polymere class can be used to determine the alignment matrix for a given pair of floating point arrays.

pol->rmsd(v1,v2,N,mtx)

Polymere::updateFull()
If the low resolution coordinates are correct calling updateFull will transform the full atom representation to match the low resolution representation.

pol->updateFull()

Potential Outline
Step1: convert the helix1 and helix2 residue positions into floating point arrays

RSIM stores two resolutions of a given RNA molecule. A low resolution representation is stores three points per residue. These three points can be accessed using the pointer to a std::vector<double> object called mols

Example:
pt1x=pol->mols[i]->x;
pt1y=pol->mols[i]->y;
pt1z=mols[i]->z;

pt2x=pol->mols[i]->bx;
pt2y=pol->mols[i]->by;
pt2z=mols[i]->bz;

pt3x=pol->mols[i]->b2x;
pt3y=pol->mols[i]->b2y;
pt3z=mols[i]->b2z;

The high resolution representation stores 27 points for each residue. These are found in the vector full_atoms. A given point can be accessed using the following:

pt1x = full_atoms[0]
pt1y = full_atoms[1]
pt1z = full_atoms[2]
...

Each residue has 27*3 entries in full_atoms.

Step2: extract the transformational matrix from the rsmd function

Step3: apply the transformation matrix to the full_atoms vector and mols[i]->x,y,z,bx,by,bz,b2x,b2y,b2z coordinates or apply matrix to low resolution coordinates and call updateFull.