# CS100M Spring 2008 Project 5 due Thursday 4/10 at 6pm

You must work either on your own or with one partner. You may discuss background issues and general solution strategies with others, but the project you submit must be the work of just you (and your partner). If you work with a partner, you and your partner must first register as a group in CMS and then submit your work as a group.

## Objectives

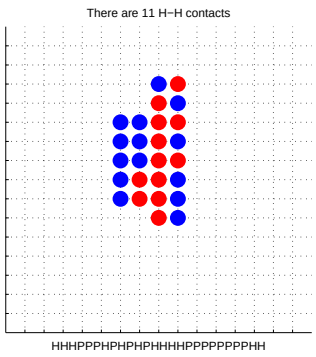
There are two questions in this project. Question 1 is given below, and Question 2 will appear in a separate document. You will learn more about graphics, matrices, cell arrays, structure arrays, and reading data files. You will do more *design* this time than in previous projects. Specifically, in Question 1, we specify the tasks that your program must accomplish, but it is up to you how to structure the program—encapsulate specific, detailed tasks as subfunctions so that the structure of your program (main function) is clear. Remember to write clear and concise comments!

## 1 Protein Folding

Protein folding is one of many difficult problems from the field of Computational Biology. A protein is basically a string where each character represents one of the 20 amino acids. The goal of the protein-folding problem is to predict the protein's 3-dimensional structure given the sequence of amino acids. This is a difficult problem, and currently, the most successful techniques don't predict structure based simply on the rules of chemistry and physics. Instead, structure is predicted by "folding" the query protein into carefully chosen known-structures (i.e., those with some sequence-similarity to the query protein) and then checking to see which has the lowest energy.

Because the prediction of protein structure is so difficult, researchers have been interested in simplified models of proteins. We'll use a version of one such simplified model, called the HP Model. In the HP Model, the 20 amino acids are reduced to just two: H (hydrophobic) and P (polar or hydrophilic). As indicated by the name, hydrophobic amino acids hate water. The preferred position for an H is in the interior of the protein, away from any surrounding water. P amino acids are happy to be anywhere; they don't care about the surrounding water. We simplify the folding so that each amino acid lies on a lattice point of the Cartesian grid. Amino acids that are adjacent along the protein string are adjacent in the grid. The folding goal becomes: Find the lattice-respecting fold that maximizes the number of adjacent H-H pairs; this is simple to count and maximizing the number of H-H contacts has the effect of "hiding" the hydrophobic amino acids. It has been shown [Berger & Leighton, RECOMB 1998] that even this simplified version of protein folding is *NP-complete* on the 3D Cartesian lattice, i.e., there is no algorithm known that can efficiently solve this problem.

For this assignment, we simplify the problem further by restricting ourselves to the 2D Cartesian grid. (The 2D version of the HP folding problem has a reasonably efficient algorithm, but this algorithm is beyond the scope of this course.) You are to write a program that uses mouse-clicks to layout an HP string on the 2D grid and then reports the number of H-H contacts. Your program does *not* try to find the best layout; instead, your program simply *evaluates a 2D layout as clicked in by a user*. Below is the figure window showing the layout clicked in by a user for a given string. The string is shown at the bottom of the figure and the top of the figure shows the number of H-H contacts.



You will submit a function `hpLayout` that accepts a *string* argument consisting of the letters 'H' and 'P'. (The example diagram was based on the function call `hpLayout('HHHPPPHPHPHPHHHHPPPPPPPHH')`). Your program is to accomplish the following tasks.

a. Display the grid. To see how to do this, take a look at the example random-walk programs (`RandomWalk2D` and `ShowRW`) that was used in an earlier lecture. A 15-by-15 grid is big enough for any example that we'll be doing. `ShowRW.m` also demonstrates how to display larger disks using the plot property `Markersize`. Another resource is Appendix A of the posted FVL text (check out the first three topics).

b. Place one amino acid (either an H or a P) on the grid for each legal mouse-click. Use the built-in function `ginput` to get mouse clicks. Recall that `[x,y]=ginput(1)` will store in variables `x` and `y` the Cartesian coordinates of one mouse click.

c. Indicate which amino acids are H and which are P in your picture. The example figure uses a blue ball for each P (polar amino acid, loves water) and a red ball for each H (hydrophobic, hates water), but you can use some other scheme if you prefer.

d. Indicate when a user has clicked illegally. Use the figure's title for such messages (use built-in function `title`). A click is illegal if it's outside the set of valid grid points, if it's not immediately adjacent to the previously placed H or P, or if it's attempting to use an already occupied grid point.

e. Indicate which amino acid is going to be placed on the grid for the next click. One way to do this is to display just the part of the amino acid string (H's and P's) that remains to be placed. Our solution shows the string at the bottom of the figure using the built-in function `xlabel`.

f. After the layout is done, display the original string of H's and P's that represent the protein. (Our solution uses `xlabel` to show the string on the bottom of the figure.)

g. Calculate and display the number of H-H contacts on the figure (as the title). Two H's are in contact if they are adjacent on the grid. Note that certain contacts must occur because the H's are adjacent in the string. For example, the string 'HHHPPPHPHPHPHHHHPPPPPPPHH' must have at least 6 H-H contacts no matter how it's laid out because there are 6 adjacent H-H pairs in the string.

Hint: Consider using one or more helper functions in your program design. For instance, it's useful to have a helper function that handles mouse-clicks. Such a function produces error messages until a legal mouse-click occurs; then it returns the coordinates of the legal click. Note that error messages are supposed to appear in the figure (we used the figures title to display error messages), not in the Command Window. Finally, remember that *design* is a non-trivial process! Take some time to design your solution; this includes coming up with an algorithm for the iterative drawing problem, specifying helper functions, and choosing helpful data structures (e.g., matrix, vector).

Just for fun, see if you can find a layout for the example protein in the figure that has more than 11 H-H contacts. One of our attempts produced 13 H-H contacts.