```
%HW 8
%Audrey and Irene
%Problem 2%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%To generate a neighboring solution, we randomly lock/unlock the legs and
%contract/release the muscles based on the temperature. With a high
%temperature, we flip more of the current legs and muscle bits to â€œexplore
%a larger range.â€  The first rule of flipping is at any interval, there must
%be two locked legs so the manduca can crawl. The second rule we made is
%that both the front and last leg cannot be locked at the same time. The
%third rule we made is that if there are two consecutive locked legs the
%muscle in between them must be contracted. The fourth rule we made is that
%at any given interval one of the muscles has to be actuated. A neighboring
%solution is defined as a solution that is one step away from the initial
%solution while satisfying the constraints we created.

%Problem 3%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%The number of bits we flip is based on the ratio between the initial and
%current temperature with respect to the number of legs/muscles. When the
%temperature is high, the ratio is also high. This means that more legs and
%muscles will flip in an interval with a high temperature as opposed to
%when the temperature is low. The number of bits we flip is the rounded
%number of the ratios computed.

%Problem 7%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%For our base case, after 10 random restarts with an initial temperature of
%80, the best score we got was 238.296715. The parameters we implemented
%were running the program through 100 iterations, setting the initial
%temperature to 80, final temperature to 15, and alpha to 0.999. In our
%base case, we also included the following constraints: there always needed
%to be two locked legs on any given interval and if there were two locked
%legs next to each other, the muscle in between them had to be contracted.
```

```
>> leg matrix
     1     0     0     0     1
     1     0     0     1     1
     1     1     0     1     1
     1     1     0     0     0
     0     0     0     0     1
     1     0     0     0     1
     1     0     0     1     0
     0     0     0     1     1
     1     0     1     1     0
     1     1     1     0     0

>> muscle matrix
   100   100     0     0
   100   100     0     0
     0     0     0     0
     0     0     0     0
     0   100   100   100
     0     0   100     0
     0   100   100   100
   100     0   100     0
   100   100     0     0
     0     0     0     0
```

In our updated or improved solution, we changed the number of iterations to 1000 to let the program have more time to explore a greater range of the solution space. We added a multiplier to the temperature ratio of both the legs and muscles in order to change the number of bits we flip at a given time interval, in other words we halved the number of bits we flipped to see how that would change the manducaâ€™s crawl. We also experimented with alpha and tried lower values such as 0.95 and higher values such as 0.9999, both of which resulted in lower distances traveled so we kept the initial alpha value of 0.999. We tried a lot of different initial and final temperature combinations ranging from 1 to 10000 using nested for loops and figured out that for our program, the lower temperatures gave us better solutions. We decided an initial temperature of around 5 and final temperature around 0.5 gave us the best solutions. Initially, we only applied two constraints to our manduca. In the updated solution, we added the following constraints: the front and back leg could not be locked at the same time on an interval and at least one muscle has to be actuated at any given time.

After 10 random restarts with an initial temperature of 3.8, our best score was 1081.720

```
bestleg =

     0     0     0    0     1
     1     0     0    0     0
     0     1     1    0     1
     1     0     0    0     0
     1     0     0    1     0
     0     0     0    1     1
     1     1     1    1     0
     0     1     0    1     1
     1     0     0    0     0
     0     0     0    1     1

bestmusc =
   100   100   100   100
     0   100     0   100
   100    0      0     0
     0    0      0   100
   100    0      0     0
     0  100    100     0
     0    0      0   100
     0  100      0     0
     0    0      0   100
     0  100      0     0
```