Numerical Analysis and Programming

Lab Worksheet #5

1. *Palindrome* A palindrome is a word that is spelled the same backward and forward, like "noon" and "redivider". Recursively, a word is a palindrome if the first and last letters are the same and the middle is a palindrome.

   (a) Write a function defined as

   ```
   def is_palindrome(s):
   ```

   that takes a string argument `s` and returns `True` if it is a palindrome and `False` otherwise using string methods.

   (b) Use this function to write a function `has_palindrome` defined as

   ```
   def has_palindrome(s, start, length):
   ```

   which determines if the substring of string `s` starting from `start` with length `length` is palindrome.

   (c) Use this function to solve the following puzzler:

   > "I was driving on the highway the other day and I happened to notice my odometer. Like most odometers, it shows six digits, in whole miles only. So, if my car had 300,000 miles, for example, I'd see 3-0-0-0-0-0.
   > "Now, what I saw that day was very interesting. I noticed that the last 4 digits were palindromic; that is, they read the same forward as backward. For example, 5-4-4-5 is a palindrome, so my odometer could have read 3-1-5-4-4-5.
   > "One mile later, the last 5 numbers were palindromic. For example, it could have read 3-6-5-4-5-6. One mile after that, the middle 4 out of 6 numbers were palindromic. And you ready for this? One mile later, all 6 were palindromic!
   > "The question is, what was on the odometer when I first looked?"

   Write a function `puzzle_solver()` which tests all six-digit numbers and return the possible numbers that satisfy these requirements.

2. *Using Lists as Stacks* The list methods make it very easy to use a list as a stack, where the last element added is the first element retrieved ("last-in, first-out"). To add an item to the top of the stack, use `append()`. To retrieve an item from the top of the stack, use `pop()` without an explicit index. It is easy to implement non-recursive version of recursive function using stacks. Implement `factorial(n)` for non-negative integer n using a stack. You may want to go back to study the recursive version of `factorial(n)`.