Donovan Guelde
dogu7023
CSCI-3104 HW 7

1. For every location n, we need to determine the maximum possible profit. So:

put locations and distances into arrays, n[ ], m[ ]

for each location n_i, iterate through the remaining distances, m[ ]. If the distance m_j - K >0, then that restaurant is a possibility, so add the corresponding profit value to array P[ ]. After iterating through m[ ], simply choose the max value in P[ ], and remove the corresponding item from m[ ].

This procedure would run in $O(n^2)$ time, as it has a nested loop structure.

2.

```
start with: a = 0, b = N-1 (we start with the first and last character of the string)
findPalindrome(string, int a, int b):
        if a == b, return 1 (this is a palindrome of length 1, 'base case')
if (S[a] == S[b]) (first,last characters of a possible palindrome)
        return 2+findPalindrome(a+1,b-1) (working our way inside)
if (S[a] != S[b]) (if not a possible palindrome, check (a+1,b) and (a,b-1) for palindromes)
        return max(findPalindrome(a,b-1),findPalindrome(a+1,b)
```

On my honor, as a University of Colorado at Boulder student, I have neither given nor received unauthorized assistance.