

CSCI 135 Software Design and Analysis, C++  
Homework 7  
Due 4/11/2014

Saad Mneimneh  
Hunter College of CUNY

**Problem 1: Palindromes**

A palindrome is a string that reads the same forward and backward when spaces are ignored. Here are example palindromes:

radar

kayak

a man a plan a canal panama

no x in nixon

110101101011

evil olive

was it a car or a cat I saw

never odd or even

For simplicity, we will assume that our alphabet consists of the following symbols  $\{'0' - '9', 'a' - 'z', ' '\}$ .

(a) Write a function called `next` that takes as parameters a string  $s$ , and two indices  $i$  and  $j$ , and returns the smallest  $k$ ,  $i < k \leq j$  such that  $s[k]$  is not a space, or -1 if such an  $k$  does not exist.

(b) Write a function called `prev` that takes as parameters a string  $s$ , and two indices  $i$  and  $j$ , and returns the largest  $k$ ,  $i \leq k < j$  such that  $s[k]$  is not a space, or -1 if such a  $k$  does not exist.

(c) Write a function called `neq` that takes as parameters a string `s`, and two indices `i` and `j`, and returns the smallest `k`,  $i \leq k \leq j$  such that `s[k]` is not a space, or -1 if such an `k` does not exist.

(d) Write a function called `peq` that takes as parameters a string `s`, and two indices `i` and `j`, and returns the largest `k`,  $i \leq k \leq j$  such that `s[k]` is not a space, or -1 if such a `k` does not exist.

(e) Write a function with the following signature:

```
void init(const char * s, int i, int j, int * k, int * l) {...}
```

or

```
void init(const char *s, int i, int j, int& k, int& l) {...}
```

that makes `k` and `l` the indices of the two middle non-space characters in `s[i...j]` (`k` could be the same as `l`), or -1 if `k` and `l` are not defined. For instance, if `s = "evil olive"`, `i = 0`, `j = 9`, then `k = l = 5`, and if `s = "no x in nixon"`, `i = 0` and `j = 12`, then `k = 6` and `l = 8`. Ideally, your function should use the four functions described above.

(f) In this part, you will use the five functions developed above in any way you find useful. Write a function called `parlindrome` that takes as parameters a string `s`, and two indices `i` and `j`, and returns true if and only if `s[i...j]` is a palindrome. Your function should work from the inside out. In other words, if `s = "evil olive"`, and `i = 0` and `j = 9`, your function will first compare 'o' to 'o', then 'l' to 'l', then 'i' to 'i', then 'v' to 'v', then 'e' to 'e', to establish that `s` is a palindrome.

(g) [optional] Using the inside out idea in (f), write a function called `longest-Palindrome` that takes a string `s` and outputs the longest palindrome in `s`. Your function should contain at most doubly nested loops. In other words, you cannot base your function on the following idea (triply nested loops):

```
for (int i=0; i<strlen(s); i=i+1)
  for (int j=i; j<strlen(s); j=j+1)
    //check if s[i...j] is a palindrome (which contains a loop)
```

Basically, we want this function to have an  $O(n^2)$  amount of work, and not  $O(n^3)$ .

(h) Write a program that allows the user to input a string and outputs whether the string is a palindrome or not, and if not output the longest palindrome in that string (this falls into the optional part).