University of Zurich^UZH

Solution

Department of Informatics
Database Technology Group
Prof. Dr. M. Böhlen

Informatik II
Exercise 1 / **Solution**

Feb 24, 2020

Introduction to C

Task 1

Solution : Iterate each element/character of the string, using some index, until you find the null character(\0). Once you find the null character, stop iterating the string and the value of index will give the length of the string.

Pseudocode

```
Algo: strLength(s)
$i \leftarrow 0$;
while $s[i] \neq$ '\0' do
    $i = i + 1$
return $i$
```

Task 2

Solution: Iterate the string form start to half of its length. In each iteration, for character at position i, compare it with the character at position i from the end. Meanwhile count the number of matched characters and if at the end they equal half the length of string then the string is palindrome otherwise not.

Alternate Solution: Whenever a mismatch is found you can terminate the loop while displaying that string in not palindrome.

Pseudocode

```
Algo: isPalindrome(s)
$c \leftarrow 0$
$sLength \leftarrow strLength(s)$
for $i = 0$ to $sLength/2$ do
    if $s[i] == s[sLength - i - 1]$ then
        $c = c + 1$
if $c == sLength/2$ then
    print ”$TRUE$”
else
    print ”$FALSE$”;
```

Task 3

Solution: You will need a nested loop to compare every integer value in array A with values of array B. When the value matches, increment the counter and keep iterating the array B until the values keep matching. However, in case of mismatch, if the value of array B is less than the value of array A, then iterate the array B until the value equal or greater than in A is found.

```
Algo: countPairs(A, nA, B, nB)
$i \leftarrow 0$
$j \leftarrow 0$
$count \leftarrow 0$
while $i < nA$ do
    while $A[i] > B[j]$ and $j < nB$ do
        $j = j + 1$
    while $A[i] == B[j]$ and $j < nB$ do
        $j = j + 1$
        $count = count + 1$
    print $(A[i], count)$
    $i = i + 1$
    $count = 0$
```

Sorting (n^2)

Task 4

Solution: Apply the insertionSort on an array and then separate the even and odd integers.

```
Algo: insertionSort(A,n)
int $position, value$
for $i = 1$ to $n$ do
    value = A[i]
    position = i
    while $position > 0$ and $A[position - 1] > value$ do
        A[position] = A[position-1]
        position = position - 1
    A[position] = value
return A
```

```
Algo: evenOddInsertionSort(A, n)
$evenCount \leftarrow 0$
$oddCount \leftarrow 0$
int $evenList[n]$
int $oddList[n]$
$A \leftarrow insertionSort(A, n)$
for $i = 0$ to $n - 1$ do
    if $A[i]\%2 == 0$ then
        evenList[evenCount] = A[i]
        evenCount = evenCount + 1
    else
        oddList[oddCount] = A[i]
        oddCount = oddCount + 1
print $evenList, evenCount, oddList, oddCount$
```