University of Zurich[UZH]

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

Informatik II
Exercise 1

Feb 18, 2020

Introduction to C

Task 1. Write a C program with a function `int strLength(char s[])` that determines the length of string `s`. Assume that all strings are at most 1000 characters long. Your program should prompt the user for an input string, read the string (terminated by a newline), and print the length of the input string to the screen. Do not use any build-in library functions to calculate the length of the string. An input/output example is illustrated below (the user input is typeset in bold):

```
Please enter a string: Hello World
String Length: 11
```

Task 2. Write a C program with a function `bool isPalindrome(char s[])` that determines whether a string is a *palindrome* or not. A palindrome is a string that reads the same backwards as forwards. You can use function `strLength(char s[])` written in Task 1 to determine the string length. Your program should prompt the user to type input string and then should print "TRUE" if it is a palindrome or "FALSE" if it isn't. An input/output example is illustrated below (input is typeset in bold):

```
Please enter a string to check if it is palindrome: wow
Result string: TRUE
```

Task 3. Consider an array `A[0...`n_A`-1]` with n_A distinct integers, and an array `B[0...`n_B`-1]` with n_B integers that might not be distinct. Arrays `A` and `B` are both sorted in increasing order. Write a program in C that reads arrays `A` and `B` and prints all pairs of the form `(v,t)`, where `v` corresponds to a value in array `A` and `t` to the number of times it appears in array `B`. Your program may not use any sorting algorithm. An input/output example is illustrated below (input is typeset in bold):

```
Values of A separated by spaces (non-number to stop): 1 2 3 end
Values of B separated by spaces (non-number to stop): 1 2 2 2 3 3 4 end
Pairs: (1,1) (2,3) (3,2)
```

Sorting (n^2)

Task 4. Write a program in C that reads an array `A[0...n-1]` with `n` integers and implements the function `int evenOddInsertionSort(int A[], int n)`. This function should return the following elements: a sorted array `E` with the even numbers in `A`, the sum of all elements in `E`, a sorted array `O` with the odd numbers in `A`, and the sum of all elements in `O`. An input/output example is illustrated below (input is typeset in bold):

Values of A separated by spaces (non-number to stop): **2 10 3 22 15 12 end**
Sorted even numbers: 2, 10, 12, 22
Sum of even numbers: 46
Sorted odd numbers: 3, 15
Sum of odd numbers: 18