Week 2 Assessment Questions

Analysis of Algorithms

1. (Algorithms and complexity)

   Develop an algorithm to determine if a character array of length *n* encodes a *palindrome*, that is, which reads the same forward and backward. For example, "racecar" is a palindrome.

   a. Write the algorithm in pseudocode.

   b. Analyse the time complexity of your algorithm.

   c. Implement your algorithm in C as a function

   ```
   bool isPalindrome(char A[], int len)
   ```

   that returns `true` if string `A` of length `len` is a palindrome, and `false` otherwise.

   Hint: The standard library `<stdbool.h>` defines the basic data type `bool` with the two values `true` (internally encoded as `1`) and `false` (= `0`).

   d. Use your solution to Exercise c. to write a program that prompts the user to input a string and checks whether it is a palindrome. Examples of the program executing are

   ```
   prompt$ ./palindrome
   Enter a word: racecar
   yes
   prompt$ ./palindrome
   Enter a word: reviewer
   no
   ```

   Hint: You may use the standard library function `strlen(char[])`, defined in `<string.h>`, which computes the length of a string (without counting its terminating `'\0'`-character).

We have created a script that can automatically test your program. To run this test you can execute the `dryrun` *program that corresponds to this exercise. It expects to find a program named* `palindrome.c` *in the current directory. You can use dryrun as follows:*

```
prompt$ 9024 dryrun palindrome
```

2. (Algorithms and complexity)

   A vector V is called *sparse* if most of its elements are 0. In order to store sparse vectors efficiently, we can use an array L to store only its non-zero elements. Specifically, for each non-zero element *V[i]*, we store an index-value pair *(i,V[i])* in L.

   For example, the 8-dimensional vector *V*=(2.3,0,0,0,-5.61,0,0,1.8) can be stored in an array L of size 3, namely L[0]=(0,2.3), L[1]=(4,-5.61) and L[2]=(7,1.8). We call L the *compact form* of V.

   Describe an efficient algorithm for adding two sparse vectors V_1 and V_2 of equal dimension but given in compact form. The result should be in compact form too, of course. What is the time complexity of your algorithm depending on the sizes *m* and *n* of the compact forms of V_1 and V_2, respectively?

   Hint: The sum of two vectors V_1 and V_2 is defined as usual, e.g. (2.3,-0.1,0,0,1.7,0,0,0) + (0,3.14,0,0,-1.7,0,0,-1.8) = (2.3,3.04,0,0,0,0,0,-1.8).