Programming report
Assignment 1 - Anagram for A&DinC

First Author (student number)
Mathijs van Maurik (s2059525)

February 14, 2017

[*About the writing style: keep it professional. Write correct sentences, check your spelling.*]

1 Problem description

General: Write a C program that ....

Input-output behaviour: [*Here you explain what the problem is. Suggestion: first give a short description in general terms, then provide a more precise problem specification, e.g. in terms of input-output behaviour.*]

2 Problem analysis

[*Here you analyze the problem: what is the essence of the problem, and how can it be solved? Useful methods are: abstraction, divide-and-conquer, and analogy. Abstraction skips the irrelevant details and focuses on the essence of the problem, which may be described in mathematical terms (sets, functions, arrays, etc.). Divide-and-conquer divides the problem in subproblems, in such a way that solution of the subproblems leads to a solution of the original problem. Analogy looks for problems that are similar to the problem at hand and for which we already have a solution. When you see more than one way to solve the problem, you may indicate this, and explain why you choose one specific solution. You may also use pseudocode (Appendix C of the lecture notes) to describe one or more algorithms that are part of the solution.*]

3 Program design

[*Here you indicate how you translate your ideas for solving the problem into a C program. The general advice here is: keep it general! Do not explain every statement in the program, but describe instead the strategic choices you made. Examples of such choices are the data structures, the functions and the programming constructs that are used in the program.*]

4 Evaluation of the program

[*Here you report about the testing and the performance of the program. Indicate (here or in the Appendix) the test sets that you have used, and the output they generated. Also indicate whether your program was accepted by Themis. Do not forget to check for memory leaks.*]

5 Extension of the program (optional)

[*In this optional section you describe any extensions you made to the program. Indicate whether you followed one of the suggested extra's, or that you came up with an extension of your own.*]

6 Process description

[*Here you describe shortly the process that led to the final code and the report. What was easy, what was difficult? Did you make interesting mistakes? What have you learned from this assignment? Also indicate who did what while working on the assignment.*]

7 Conclusions

[*Here you formulate your conclusions. Does your program solve the problem? Is it efficient? It is optimal?*]

8 Appendix: program text

[*Here you provide the program text. It should correspond exactly with the final program that you submitted to Themis. If this is not the case, motivate why there are differences and indicate clearly what they are. The program should contain appropriate comments so as to improve readability. The program should satisfy the criteria given in Appendix D of the lecture notes. This LaTeX file contains an example how to include program files.*]

Listing 1: `prog.c`

```
1  /* file: anagram.c                                          */
2  /* author: A.M van Maurik (email: mathew9753@gmail.com) */
3  /* date: 14-02-2017                                         */
4  /* version: 1.00                                            */
5  /* Description:
6  */
7
8
9
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <math.h>
13
14
15 int main(int argc, char *argv[]) {
16    // Variables
17    int n;
18    int m;
19
20    // The first integer input is the number of sentences in the table.
21    scanf("%d",&n);
22
23    //Looping through n sentences.!  Sentence ends with "." Full stop.
24
25    // The second integer input is the number of test sentences.
26    scanf("%d",&m);
27
28    //Looping through m test sentences.! Sentence ends with "." Full stop.
29
```

```
30
31
32      // PROGRAM! FIRST DETERMINE WICH SENTENCES HAVE THE SAME LENGTH (WITHOUT
            SPACING)
33
34
35      // Testing everything.
36      printf("Hello␣World!\n");
37      printf("table␣sentences:␣%d␣and␣testing␣sentences:␣%d\n", n, m);
38
39      return 0;
40  }
```

9 Appendix: test sets

[*When you use large test sets for the evaluation of the program, you can give them here.*]

10 Appendix: extended program text (optional)

[*In this optional section you provide the text of the extended program. The same criteria apply as for the program text.*]