Assignment 3

COMP9021, Session 1, 2013

1 Aim of assignment and general description

The aim of the assignment is to analyse English sentences and solve logical puzzles. The assignment will make you practice the use of pointers, command line arguments, and dynamic memory allocation.

Raymond Smullyan has designed many puzzles involving Knights and Knaves. Knights always tell the truth, whereas Knaves always lie. We refer to Knights and Knaves as Sirs. A puzzle, which is a set of English sentences, involves a finite number of Sirs. Solving the puzzle means:

- determining the names of all Sirs involved in the puzzle;
- determining solutions to the puzzle, where a solution qualifies each Sir as either a Knight or a Knave.

Some puzzles have no solution, others have a unique solution, and others have at least 2 solutions. The following is an example of a puzzle with a unique solution.

```
One evening as you are out for a stroll, you walk by a doorway labeled no normals
allowed.  Some people are talking inside.  Curious, you listen, and you hear
Sir Paul who says:  "all of us are Knaves." "Exactly one of us is a Knight,"
replies Sir Jenny.  As for Sir John, who is also inside, he just keeps quiet.
Who is a Knight, and who is a Knave?
```

The Sirs involved in this puzzle are Sir Jenny, Sir John, and Sir Paul. The unique solution is given by Sir Jenny being a Knight, Sir John being a Knave, and Sir Paul being a Knave.

2 Detailed description

2.1 Syntax of puzzles

A sentence starts with a capital letter and ends in a full stop, an exclamation mark, or a question mark, possibly followed by closing double quotes. Sir, Sirs, Sir names, Knight and Knave always start with a capital letter, and no other word inside a sentence is capitalised. A sentence in a puzzle contains at most one part enclosed between double quotes. When a sentence contains one part enclosed between double quotes, the part outside the double quotes contains a single occurrence of the form `Sir` *`Sir_Name`*, and what occurs between the double quotes is something said by `Sir` *`Sir_Name`*. A sentence that contains no part enclosed between double quotes might refer to a number of Sirs, always in the form `Sir` *`Sir_Name`*, or `Sirs` *`Sir_Name_1`* `and` *`Sir_Name_2`*, or `Sirs` *`Sir_Name_1`*`,` *`Sir_Name_2`*`,` ... `and` *`Sir_Name_n`*, where $n \geq 3$, and *`Sir_Name_1`*, ..., *`Sir_Name_n`* are pairwise distinct.

What is between double quotes is a sentence in one of the following forms, ending in either a comma, a full stop, an exclamation mark, or a question mark:

- At/at least one of *Conjunction_of_Sirs*/us is a Knight/Knave
- At/at most one of *Conjunction_of_Sirs*/us is a Knight/Knave
- Exactly/exactly one of *Conjunction_of_Sirs*/us is a Knight/Knave
- All/all of us are Knights/Knaves
- I am a Knight/Knave
- Sir *Sir_Name* is a Knight/Knave
- *Disjunction_of_Sirs* is a Knight/Knave
- *Conjunction_of_Sirs* are Knights/Knaves

where:

- *Disjunction_of_Sirs* is in one of the following forms:
  - *Sir_1* or *Sir_2*
  - *Sir_1*, *Sir_2*, ... or *Sir_n* ($n \geq 3$)
- *Conjunction_of_Sirs* is in one of the following forms:
  - *Sir_1* and *Sir_2*
  - *Sir_1*, *Sir_2*, ... and *Sir_n* ($n \geq 3$)
- *Sir_1*, ..., *Sir_n* are pairwise distinct expressions of the form Sir *Sir_Name* or I.

2.2 Input and output of programs

Your program will accept the sentences that make up a puzzle as command line arguments. Practically, assuming that `solve` is the name of the executable file and the puzzle is stored in a file `test.txt`, your program will be run with the command

```
solve $(cat test.txt)
```

No assumption should be made on the number of English sentences provided as input, nor on the length of a sentence, nor on the length of a Sir name, nor on the number of Sirs involved in the puzzle.

Your program should:

- output the Sirs involved in the puzzle in lexicographic order;
- output whether or not there is a solution, and in case there is one, whether the solution is unique;
- outputs the solution in case a unique solution exists, with all Sirs being qualified as either Knight or Knave in alphabetical order.

2.3 Sample outputs

Here are a few tests together with the expected outputs of your program for the two file tests given above. The output of your program should be EXACTLY in accordance with the following outputs, WITHOUT extra blank lines, etc. Outputs of your program will be matched against expected outputs LINE FOR LINE.

```
$ cat test1.txt
I have just seen Sirs Sanjay and Eleonore!
"I am a Knave," whispered Sir Eleonore.
Who is a Knight and who is a Knave?
$ solve $(cat test1.txt)
The Sirs are: Eleonore Sanjay
There is no solution.
```

```
$ cat test2.txt
I have just met Sirs Frank, Paul and Nina.
Sir Nina said: "I am a Knight," but I am not sure
if that is true. What do you think?
$ solve $(cat test2.txt)
The Sirs are: Frank Nina Paul
There are 8 solutions.
```

```
$ cat test3.txt
Yesterday, I visited Sirs Andrew and Nancy. I asked Sir Andrew
who he was, and he answered impatiently: "Sir Nancy and I
are Knaves!" Then I met Sir Bill who introduced me to his wife
and told me: "at least one of Sir Hilary
and I is a Knave." Should I trust them?
$ solve $(cat test3.txt)
The Sirs are: Andrew Bill Hilary Nancy
There is a unique solution:
Sir Andrew is a Knave.
Sir Bill is a Knight.
Sir Hilary is a Knave.
Sir Nancy is a Knight.
```

3 Assessment and submission

3.1 Assessment

Up to eight marks will reward correctness of solutions by automatically testing your program on some tests, all different to the provided examples. Read carefully the part on program output to maximise your chances of not failing some tests for stupid reasons.

Up to one mark will reward good formatting of the source code and reasonable complexity of the underlying logic as measured by the level of indentation of statements. For that purpose, the `mycstyle`

script will be used, together with your customised `style_sheet.txt`, that you have to submit, and in which **Maximum level of indentation has to be set to 5** for this assignment. If the script identifies problems different to excessive indentation levels then you will score 0 out of 1. If the script identifies excessive indentation levels (which means that at least one line exhibits at least 6 indentation levels), then you will score 0.5 out of 1. If the script identifies no problem then you will score 1 out of 1. If your program attempts too little and contains too little code then the `mycstyle` script won't be used and you will score 0 out of 1.

Up to one mark will reward good comments, good choice of names for identifiers and functions, readability of code, simplicity of statements, compactness of functions. This will be determined manually.

Late assignments will be penalised: the mark for a late submission will be the minimum of the awarded mark and 10 minus the number of full and partial days that have elapsed from the due date.

3.2 Submission

The `main()` procedure of your program should be stored in a file called `solve.c`. Your program can contain other other `.c` and `.h` files. You are encouraged to make your program modular and split it into different files.

Go through `assignment_checklist.pdf` and make sure that you can tick all boxes (or at the least, are aware that you should tick them all). Then upload your files (the source code of your program and your customised `style_sheet.txt`) using WebCMS. Assignments can be submitted more than once: the last version is marked. Your assignment is due by June 9, 11:59pm.

3.3 Reminder on plagiarism policy

You are permitted, indeed encouraged, to discuss ways to solve the assignment with other people. Such discussions must be in terms of algorithms, not C code. But you must implement the solution on your own. Submissions are routinely scanned for similarities that occur when students copy and modify other people's work, or work very closely together on a single implementation. Severe penalties apply to a submission that is not the original work of the person submitting it.