Coursework

This coursework is inspired by the bAbI project of Facebook AI Research (https://research.fb.com/projects/babi/) which aims to improve automatic text understanding and reasoning and evaluate it through question answering. Originally it involves machine learning research, but your solution should be rule-based.

You are given:

1) a set of 6 tasks, each of which is focused on a particular aspect of text understanding: chaining facts, simple induction, deduction, etc. Each task is represented by:

- a text file containing short statements (an input file that should be taken by your programme as an argument, e.g. *task1.txt*), and
- a list of questions that the programme should read one by one from the standard input and output short answers based on the knowledge from the input file. The questions for all tasks are gathered in *questions.txt*. There you can also see the format of answers. The answer "don't know" is possible in all tasks if the knowledge from the input is not enough, except tasks 1 and 4.

2) a test file *test.txt* for validation of your programme. The questions and answers for the test are also in *questions.txt*.

Statements in the input files describe facts concerning some events, spatial attitudes or temporal relations. In a single file they form a "story", so that the information about somebody/something can change through the development of the "action". For example, this can look like:

```
John is in the garden
Bob is in the office
John picked up the football
Bob went to the kitchen
```

To answer questions, the programme might need to use information from one or several statements (supporting facts):

```
Where is John ? A: garden
Where is the football ? A: garden
Where was Bob before the kitchen ? A: office
```

Based on the types of questions and information in input files, the tasks are divided into the following:

1. *Yes/No/Maybe Questions*. Simple questions based on a single supporting fact. Lack of information here results into "maybe", not "don't know".

```
John moved to the garden
Daniel went to the bathroom
John went to the hallway
Is John in the garden ? A: no
Is Daniel in the bathroom ? A: yes
Is Mary in the kitchen ? A: maybe
```

2. *Two Supporting Facts*. Two statements have to be chained to answer the question.

```
John is in the playground
John picked up the football
Bob went to the kitchen
Where is the football ? A: playground
Where is the milk ? A: don't know
```

3. *Counting.* Calculation of the number of objects with a certain property. All numbers are 0 in the beginning.

```
Daniel picked up the football
Daniel dropped the football
Mary handed the milk to Daniel
Daniel took the apple
How many objects is Daniel holding ? A: 2
```

4. *Simple Negation and Indefinite Knowledge*. Dealing with supporting facts that imply a statement is false or describe possibilities. Lack of information results into "maybe".

```
Sandra travelled to the office
Fred is no longer in the office
John is either in the bedroom or the bathroom
Is Fred in the office ? A: no
Is Sandra in the office ? A: yes
Is John in the bedroom ? A: maybe
Is John in the office ? A: no
Is Mary in the garden ? A: maybe
```

5. *Time Reasoning*. Finding out the order of events.

```
Mary went to the park
Mary journeyed to the school
Mary is no longer in the school
Where was Mary before the school ? A: park
```

6. *Path Finding*. Given the description of various locations, the programme should output the path between them.

```
The office is east of the hallway
The kitchen is north of the office
The garden is west of the bedroom
The office is west of the garden
The bathroom is north of the garden
How do you go from the kitchen to the garden ? south, east
```

The test file contains a single "story" and involves the relations from all tasks in a mixed order. The actors (e.g. "John"), objects ("playground") and vocabulary are the same as in the training input files. So, obviously, your programme should be able to understand not only the facts from the input files, but any statements of the same types with shuffled actors and objects.

Another test file will be used to evaluate your programme. You are of course free to make up your own test stories and questions for testing. You do not need to invent more relations than those in the input files or handle contradictions or wrong input resulting in parse errors.

To make the implementation elegant, it is highly recommended to write a context-free-grammar, which parses input statements and questions (one grammar for all tasks). A possible way is to write a BNF grammar and use BNF Converter (http://bnfc.digitalgrammars.com/tutorial.html). You may use other parser libraries, if you want.

Solutions not using grammars will get a lower mark (maximum 2).

Submit a zip-file *SurnameName_coursework.zip* containing:

1. the main module *SurnameName_coursework.hs,*
2. the grammar and all modules that are generated out of it and imported by the main module (if you use BNFC), or other relevant files,
3. any text comments (especially if you failed to solve some task or wish to leave some feedback).