Moving from language to representation

You will be building a simple parser that takes sentences in English as input and produces the queries that can be handed to Ask to get you the information you need.

You will build a parser that uses a file of language patterns to make its decisions. This means that as new patterns are added, you will never make changes at the code level.

You will need to be able to manage questions of the following type:

What is the <feature> of <object>.

- What is the color of block17? -> (color block17 ?x)
- What is the shape of block17? -> (shape block17 ?x)
- What is the size of block17? -> (size block17 ?x)
- What is block17? -> (inst block17 ?x)

What <feature> is <object>.

- What color is block17? -> (color block17 ?x)

What <object type> are <feature>.

- What blocks are blue? -> ((inst ?x block) (color ?x blue))
- What spheres are big? -> ((inst ?x sphere) (size ?x big))

What <object type> is <feature>.

- What block is blue? -> ((inst ?x block) (color ?x blue))
- What sphere is big? -> ((inst ?x sphere) (size ?x big))

What is <relationship> <object>.

- What is on block17? -> (on ?x block17)
- What is under block17? -> (under ?x block17)
- What is bigger than block17? -> (bigger ?x block17)

Where is <object>.

The interesting thing is that an object can be many "places." If can be ON, UNDER or IN other objects so you need to look for all of these.

-> (on block17 ?x)
-> (under block17 ?x)

-> (on block17 ?x)

And, the answers here are going to be full statements (on block17 table) rather than just a feature “blue.” And, there may be multiple answers in that a block may be under another block and on the table.
The strategy here should be to start off with a way to represent the elements that you will need to recognize, the pattern(s) that you need to build, and the form of the answer.

For recognizing the pattern you could build a simple grammar:

Feature Question 1: ”What”, “is”, FeaturePhrase, FeaturePrepPhrase.

FeaturePhrase: “the”, Feature

Feature: Color|Size|Shape

FeaturePrepPhrase: “of”, Name

Name: String

And then state that the query you build will be:

(<Feature> <Name> ?x)

And the answer will be:

?x

Pulled together, the first “What” question would be:

Feature Question 1:
    Pattern: ”What”, “is”, FeaturePhrase, PrepPhrase.
    Query: (<Feature> <Name> ?x)
    Answer ?x

While the second would be:

Feature Question 2:
    Pattern: ”What”, Feature, “is”, Name.
    Query: (<Feature> <Name> ?x)
    Answer ?x

So once the pattern matches:
    What is the color of block17. Or What color is block17
The program builds:

```
(color block17 ?x)
```

Once you get your bindings than the “?x” can be replaced with whatever matched it.

The goal is to process a file of questions and for each one build out the query that will go against your KB using Ask. Then the bindings will be used to instantiate the answer.