Harry James Hall – C3550862

Leeds beckett university  Headingley Campus

mEDICAL DIAGNOSIS SYSTEM FOR DISEASES OF THE IMMUNE SYSTEM

# Abstract:

Immune diseases can be divided into into two broad categories: Autoimmune where the immune system attacks the body; and immune dysfunction where the immune system cannot properly defend the body against pathogens.

Auto-immune diseases include things like Lupus, Allergies, and Paraneoplastic syndromes.

Immune dysfunction includes diseases like HIV and AIDS caused by environmental factors and diseases like Chronic Granulomatosis Disease which are caused by genetics.

This system aims to help doctors or medical professionals diagnose both types of disorder based on a patients symptoms or test results.

Ideally this system would be designed to be easy to use by doctors with limited computer literacy, but this might not be achievable using only Prolog as the language the system is written in as it appears to have many limitations for making interfaces, even something like a CLI.

## Aims and objectives:

1. Provide a system that is useful in medical diagnosis to doctors looking for immune related diseases such as specialists
2. Make the system easy enough to use for doctors with minimal training

# Literature review and similar systems:

There is a medical expert system based on IBM’s Watson. IBM’s Watson is a AI system that was famous for answering questions on the American TV show Jeopardy.

It has the ability to understand questions given to it in English and answer them using information from the internet and it’s knowledge base. It also has the ability to score answers that it finds based on available evidence to determine how likely the answer is to be correct.

<https://www.nist.gov/system/files/documents/healthcare/NIST-Siegel_PART-3.pdf>

A lot of modern medical diagnostic systems use something called fuzzy logic along with neural networks.

<https://ieeexplore.ieee.org/document/9279211>

# Design:

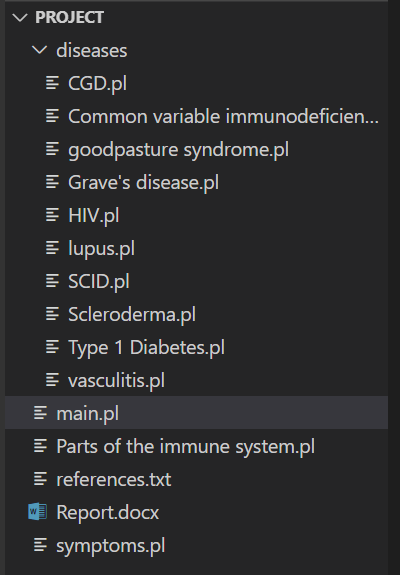
The system will first ask the doctor for a list of symptoms.

If the system can find diseases that match all of the symptoms it will display them, otherwise it will display diseases that match only some of the symptoms presented by the patient.

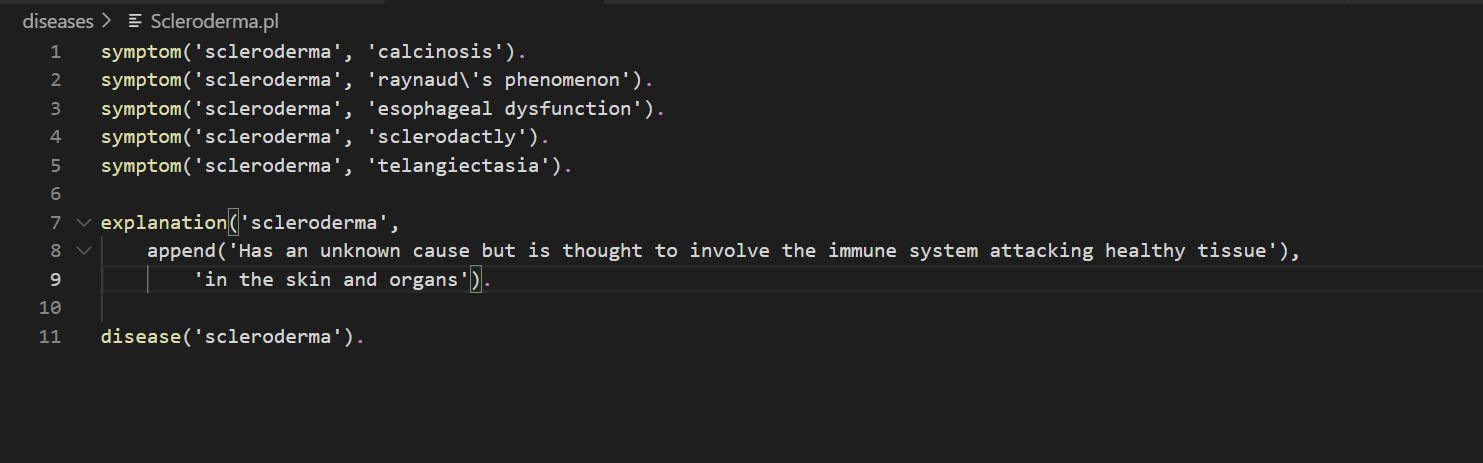
# Implementation:

The system stores it’s knowledge in separate files to the main logic.

The knowledge for each disease is stored in the diseases sub directory.

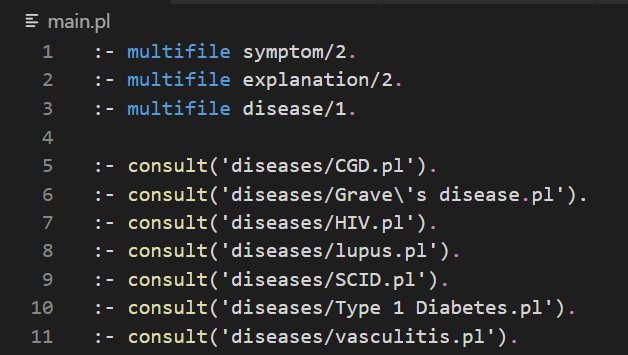


Each file has a list of symptoms stored in the symptom predicate:



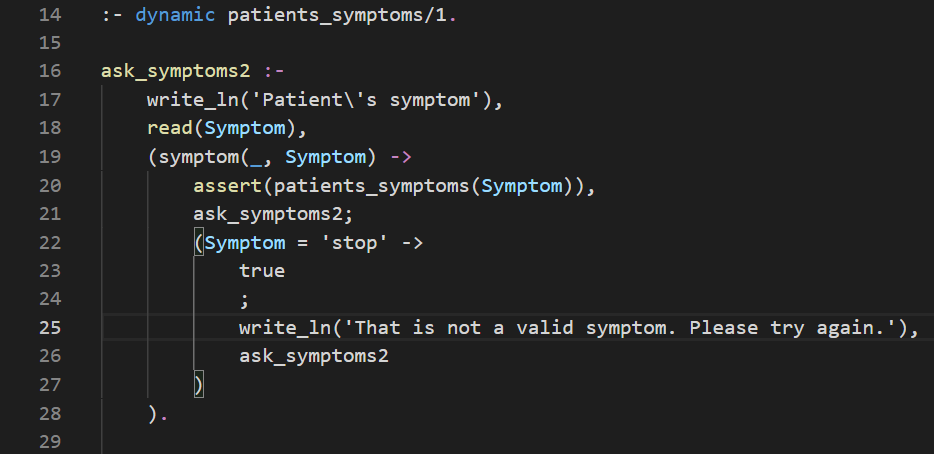
This also includes an explanation of the disease that can be printed out and the diseases name.

All of the disease files are consulted by the system at the beginning of main.pl.



The predicates symptom, explanation, and disease have to be declared as multifile predicates because they are split across multiple files otherwise it gives an error.

The program asks the doctor about the symptoms the patient has using this predicate:

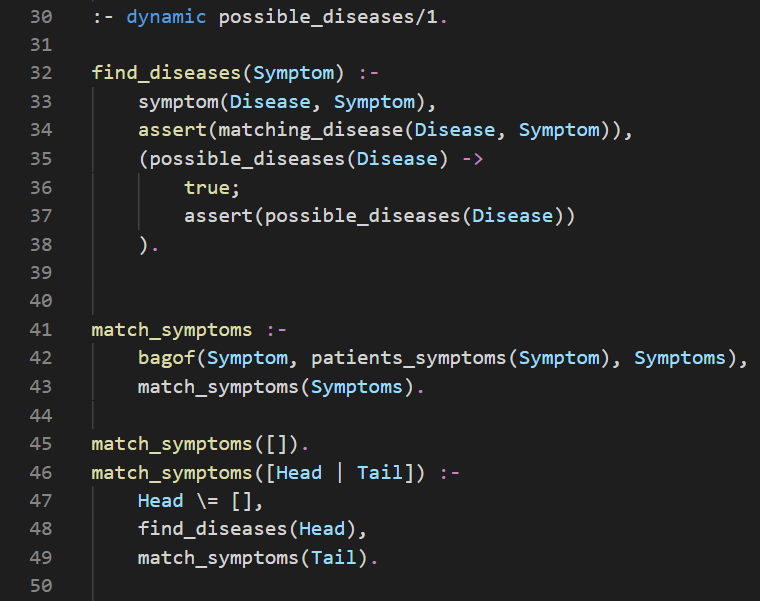


It stores the symptoms in a predicate called patients\_symptoms/1 using assert.

It uses recusion to ask for more than one symptom and stops when someone types ‘stop’.

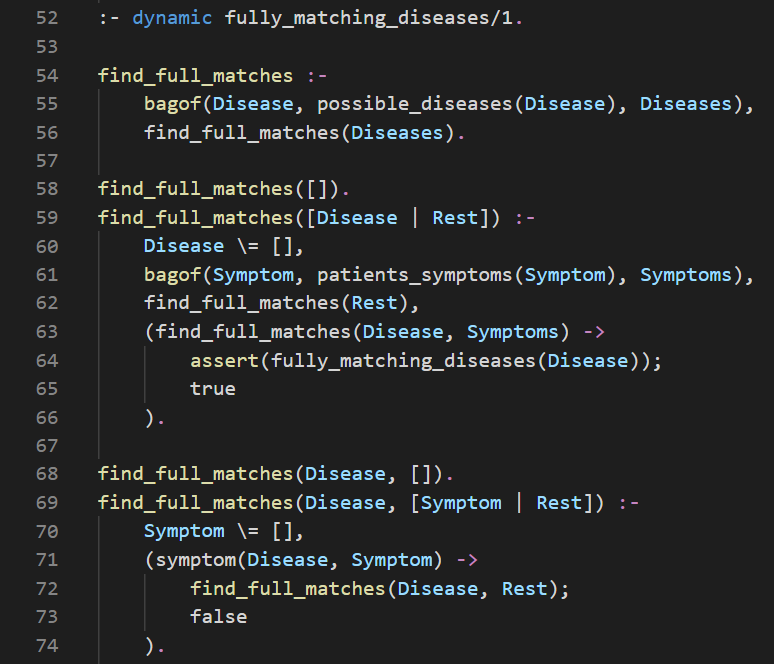
It does this using an if statement with ->.

Diseases which match some symptoms are then found using these predicates:



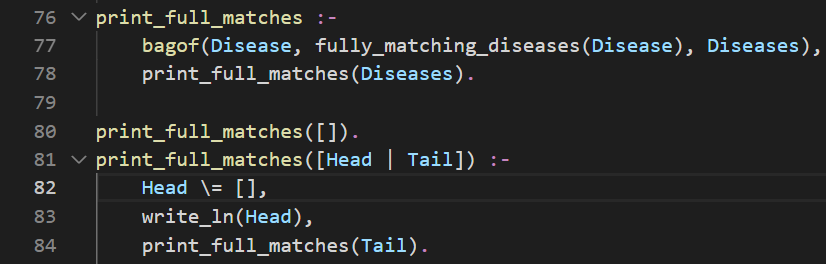
This predicate works by going through all the patients symptoms recursively and finding and trying to find a matching disease symptom pair for each one.

The find\_full\_matches finds diseases that can account for all of the symptoms.



It does this by getting a list of all the possible diseases found by the previous predicates using bag/3. It then gets a list of symptoms and checks them against the current disease before moving onto the next disease. It does this all this using recursion. If it finds a disease where all of the symptoms match it adds it to the fully\_matching\_diseases predicate using assert/1.

Next is the print\_full\_matches predicate:



The first term of the predicate gets a list of all the fully matching diseases using bag/3 and then calls the other terms of the predicate.

The third term prints out one of the matching disease and recurses on the rest. The second term ensures that the predicate terminates when there are not more fully matching diseases left in the list.