1 The Module `anagram.pl`

```
msort_string_lower(+String:string, -Lower:string,
          -Sorted:string)
```
anagram.pl

Sort the list of characters of `String` converted to lower case, and unify the results with `Sorted` and `Lower`, respectively.

```
:- module(anagram, [is_anagram/2, anagram/3]).
msort_string_lower(String, Lower, Sorted) :-
    string_lower(String, Lower),
    atom_chars(Lower, Chars),
    msort(Chars, Sorted).
```

```
is_anagram(?Word, ?Candidate)
```
anagram.pl

True if `Word` is an anagram of `Candidate`.

```
is_anagram(Word, Candidate) :-
    msort_string_lower(Word, WordLower, Sorted),
    is_anagram(WordLower, Sorted, Candidate).
```

```
is_anagram(?WordLower:string, ?Sorted:list(atom),
          ?Candidate:string)
```
anagram.pl

True if `Candidate` is an anagram of the lower case string `WordLower` where `Sorted` is assumed to be the sorted list of characters of `WordLower`, i.e., `msort_string_lower(Word, WordLower, Sorted)`.

```
is_anagram(WordLower, Sorted, Candidate) :-
    msort_string_lower(Candidate, CandidateLower, Sorted),
    \+ CandidateLower == WordLower.
```

```
anagram(+Word:string, +Candidates:list(string),
        -Anagrams:list(string))
```
anagram.pl

Filter elements `Candidate` of `Candidates` for which `is_anagram(Word, Candidate)` succeeds. True if `Anagrams` contains those elements.

```
anagram(Word, Candidates, Anagrams) :-
    msort_string_lower(Word, WordLower, Sorted),
    include(is_anagram(WordLower, Sorted), Candidates, Anagrams).
```