CO884, Assessment 2, Prolog

Marks: 20 marks per question.

In this assessment we are trying to encrypt and decrypt strings using some very basic cryptographic ciphers. Useful methods for this purpose are the library routines `string_codes/2` and `char_code/2` which translate (back and forth) between characters and numbers (char_code), or strings and lists of numbers (string_code). For example, we have `string_codes("hello", [104, 101, 108, 108, 111])`, because these numbers are the ASCII codes for the characters 'h', 'e', 'l', 'l', and 'o'.

A Caesar cipher encrypts a text by shifting all characters a fixed distance, wrapping around when the end of the alphabet is reached. We assume an ASCII alphabet with 128 characters. For example, Caesar-encryping "hello" by 1 gives us "ifmmp", encrypting it by 110 gives "VSZZ]".

1. Write a predicate **encode_list/3** such that the plaintext (first parameter) shifted by N characters (second parameter) modulo 128, gives the ciphertext (third parameter). Here, both the plaintext and the ciphertext are in the form of a list of numbers. For example, encode_list([100,63,64],78,[50,13,14]) would be true.
2. Write the predicate **encode_string/3** that does the corresponding thing, but for *strings* rather than lists of numbers. For example, we would have encode_string("hello",2,"jgnnq").
3. Write the corresponding predicates **decode_list/3** and **decode_string/3** that decrypt an encryption by reversing the shift, e.g. decode_string("jgnnq",2,X) should set X to "hello". Notice that because of the way arithmetic in Prolog works it will in general not work just to swap the arguments around.

In cryptanalysis, we are trying to find out how a text is encrypted, in this case, by how many characters the ciphertext is shifted.

4. Write a predicate **shift_distance/3**, where the first and second parameter are the ciphertext, and the plaintext, respectively (as strings). The third (output) parameter is the shift distance. So, shift_distance("jgnnq","hello",X) should succeed with setting X to 2, but shift_distance("jgnnq","peter",_) should fail.
5. Code breakers would in general not know the plain text. But they might know a bunch of words the plaintext *might* be. Write a predicate **caesar_candidate/4**, such that its first parameter is a list of plaintexts (as strings), the second a ciphertext, and the output parameters (3 and 4) are the shifting distance and the corresponding plaintext. The predicate might succeed multiple times, or not at all. But when it succeeds the plaintext must be one of the suggested texts in the first list. For example, caesar_candidate(["peter","hello"],"jgnnq",N,PT) should succeed with N=2 and PT="hello".