CS 386 Artificial Intelligence Lab
Practice Problems

The code below can be used to test for prime numbers.

```
is_prime(2).
is_prime(3).
is_prime(P):-integer(P), P > 3, P mod 2 =\= 0, \+ has_factor(P,3).

has_factor(N,L) :- N mod L =:= 0.
has_factor(N,L) :- L * L < N, L2 is L + 2, has_factor(N,L2).
```

Write the following predicates (sample output shown).

1. `maxNum(N,Ans)` where `Ans` is the largest number obtained by rotating the digits of `N`.

```
?- maxNum(21816, Ans).
Ans = 81621
```

2. `prvCir(N1, N2)` where `N2` is the largest circular prime number less or equal to `N1`

```
?- prvCir(721,Ans).
Ans = 719.
```

3. A number `n` is called abundant if the sum of all the divisors of `n` (including 1, but not `n`) is more than `n`. Below 25, there are 4 abundant numbers 12, 18, 20, 24 summing to 74. Implement `sumAb(N, Ans)` which returns the sum of all abundant numbers less than N.

```
?- sumAb(25, Ans).
Ans = 74.
```

4. Implement `sset(M, N, L, Ans)` which returns the number of subsets of `L` each containing `M` elements adding up to `N`. You can assume elements of `L` are distinct and positive (i.e. L is a set of integers greater than 0).

```
?- ssetN(3, 10, [7,1,6,2,5,4,3], Ans)
Ans =  4.
```

`Explanation : [[2,5,3],[1,5,4],[1,6,3],[7,1,2]] are the 4 subsets of size 3 adding to 10 in L.`