SUBSTITUTE PROBLEM FOR PROBLEM 2

part 1

Extend your program for assignment 2 to be able to process an if-then-else statement. To support this functionality, the EBNF given in assignment 2 has been slightly extended and modified. Everything that's not listed underneath remains the same as given in assignment 2.

EBNF:

----

expression = subexpression | if_expression;

An expression is a subexpression or an if-then-else statement.

subexpression = term { additive_operator term } ;

A subexpression is a term, followed by zero or more terms. All terms are separated by an additive-operator.

if_expression = '%if' boolean_expression 'then' complex_factor 'else' complex_factor ;

boolean_expression = '(' expression boolean_operator expression ')' ;

A boolean expression is two expressions separated by a boolean operator, within parenthesis.

boolean_operator = '=' | '<' | '>' ;

A boolean operator is '=', '<', or '>'.

There are three boolean operators:

'=' : An expression 'A = B' evaluates to:

'TRUE' if all elements that are in set A are in set B and vice versa.

'FALSE' otherwise

'<' : An expression 'A < B' evaluates to:

'TRUE' if all elements that are in set A are in set B.

'FALSE' otherwise

'>' : An expression 'A > B' evaluates to:
    'TRUE' if all elements that are in set B are in set A.
    'FALSE' otherwise

An if expression: 'if (A) then B else C' evaluates to:
    B if A is TRUE
    C if A is FALSE

For example the input:
```
?%if ({1,2,3} < {1,2,4}) then ({1}) else ({0})
```
Should print:
```
0
```

And the input:
```
a = %if ({1,2,3} > {1}) then ({1} + {2}) else ({5})
?a
```
Should print:
```
1 2
```

part 2
For this part, you have to implement set-comprehension. The EBNF will need to be modified in the following manner:

set = '{' row_natural_numbers '}' ;
    A set is a row of natural numbers between accolades.

row_natural_numbers = [natural_number_term { ',' natural_number_term }] ;
    A row of natural numbers is empty or a summation of one or more natural number terms separated by commas.

natural_number_term = natural_number ['…' natural_number] ;
    A natural number term is either one natural_number or a range of natural numbers.

Consider the input {start..end}, where "start" and "end" may be any two natural numbers.
The range thus described should add all the natural numbers in the interval from "start" to "end" to the set.

If "start" <= "end" the range should return the natural numbers contained in the interval [start, end]
If "start" > "end", the range should return the empty set.

For example the following input:

```
?{1..5}
```

Should print

```
1 2 3 4 5
```

The input:

```
?{4..7, 10, 1..2}
```

Should print:

```
1 2 4 5 6 7 10
```

And the input:

```
?{4..1}
```

Should print:

```
<empty line>
```