Nama : Adinda Aulia Sari

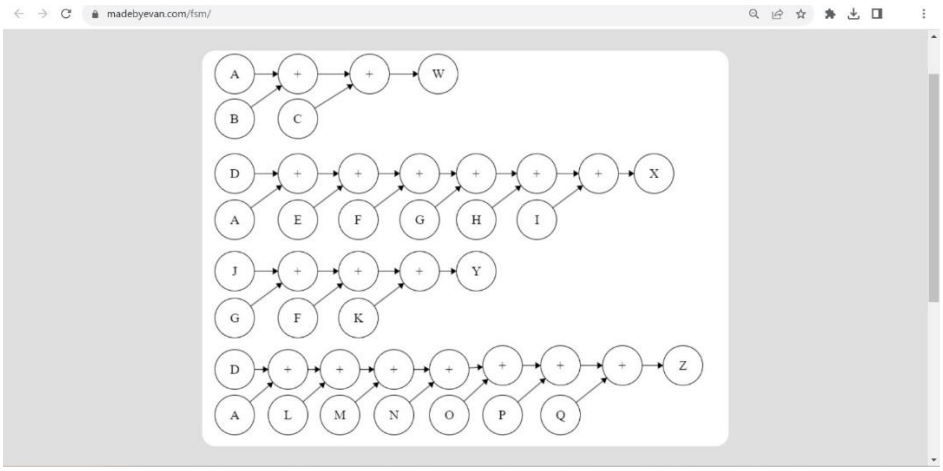
NPM : 2117051018

Kelas : CD

Latihan Chaining

A. Daftar Aturan

No	Aturan (*Rule*)
1	*IF* Sakit kepala *is True* *AND* Keluar Cairan *is True* *AND* Ada tanda-tanda radang diliang telinga *is True* *THEN* Penyakit Otitis Eksterna
2	*IF* Demam *is True* *AND* Sakit kepala *is True* *AND* PUS dan di meatus media *is True* *AND* Hidung tersumbat *is True* *AND* Hidung meler *is True* *AND* Nyeri pipi dibawah mata *is True* *AND* Selaput lendir merah dan bengkak *is True* *Then* Sinusitis
3	*IF* Bersin-bersin *is True* *AND* Hidung meler *is True* *AND* Hidung tersumbat *is True* *AND* Lendir di tenggorokan *is True* *Then* Renitis Non – Alergika
4	*IF* Demam *is True* *AND* Sakit kepala *is True* *AND* Nyeri saat berbicara atau menelan *is True* *AND* Sakit pada telinga *is True* *AND* Pembengkakan kelenjar getah bening *is True* *AND* Tenggorokan gatal *is True* *AND* Adanya tonsil yang membengkak *is True* *AND* Suara serak *is True* *Then* Farangitis (Radang Tenggorokan)



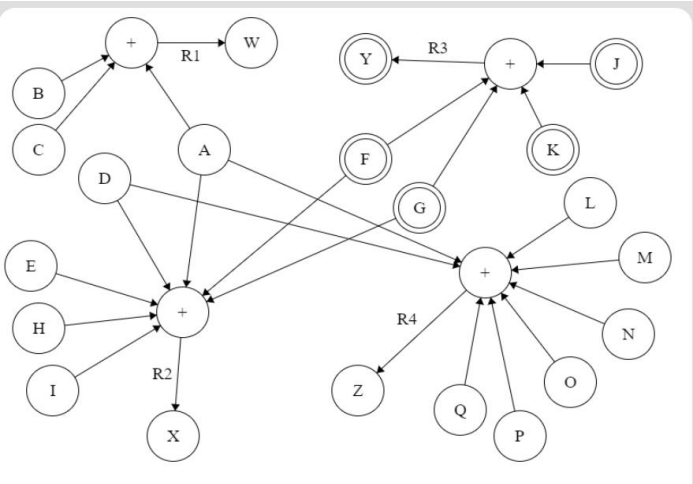
No	Aturan (rules)
R1	IF A & B & C THEN W
R2	IF D & A & E & F & G & H & I Then X
R3	IF J & G & F & K & Then Y
R4	IF D & A & L & M & N & O & P & Q Then Z

Fakta dan query :

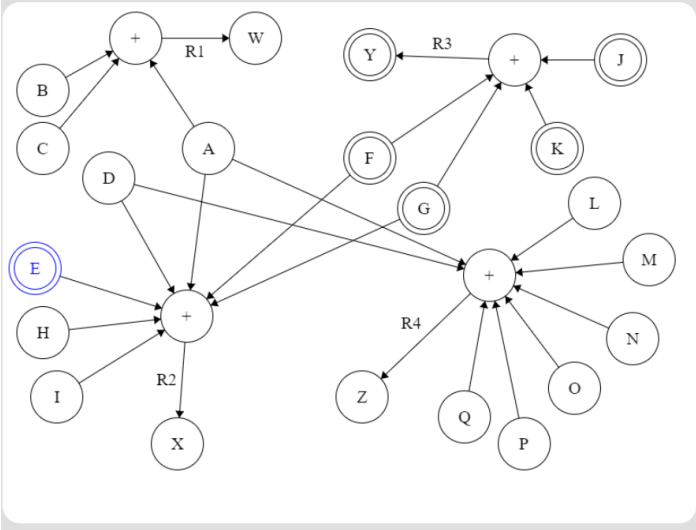
1. Fakta adalah J,G,F dan K bernilai true
   Pertanyaan (query) : apakah Y bernilai true?
2. Fakta adalah J,G,F, K, dan E bernilai true
   Pertanyaan (query) : apakah X bernilai true?

B. Graf

1. J, G, F dan K bernilai true

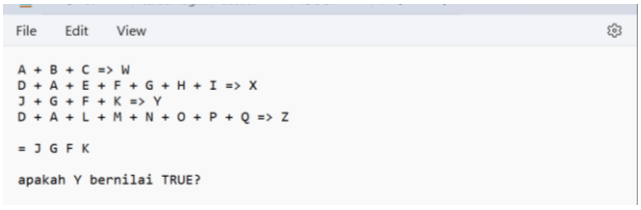


2. J, G, F, K DAN E bernilai true



C. Tampilan inputan program

1. Fakta : J,G,F,K Query : Y



2. Fakta : J,G,F,K,E Query : X

```
A + B + C => W
D + A + E + F + G + H + I => X
J + G + F + K => Y
D + A + L + M + N + O + P + Q => Z

= J G F K E

apakah X bernilai TRUE?
```

D. Tampilan output program

1. Fakta : J,G,F,K Query : Y

```
(sispak) C:\venv\sispak\Scripts>python C:\smt5\expert-system\main.py C:\smt5\expert-system\input.txt
C:\smt5\expert-system\expert_system\Tree.py:148: SyntaxWarning: "is" with a literal. Did you mean "=="?
  if self.atoms.__len__() is 0:
C:\smt5\expert-system\expert_system\Node.py:77: SyntaxWarning: "is not" with a literal. Did you mean "!="?
  ret = fixed_ret if fixed_ret.__len__() is not 0 else unfixed_ret
C:\smt5\expert-system\expert_system\Node.py:78: SyntaxWarning: "is not" with a literal. Did you mean "!="?
  if ret.__len__() is not 0:
C:\smt5\expert-system\expert_system\Node.py:84: SyntaxWarning: "is not" with a literal. Did you mean "!="?
  is_fixed = True if fixed_ret.__len__() is not 0 else False
Y resolved as True
```

2. Fakta : J,G,F,K,E Query : X

```
(sispak) C:\venv\sispak\Scripts>python C:\smt5\expert-system\main.py C:\smt5\expert-system\input.txt
X resolved as False
```

E. Penjelasan alur kerja program

Program ini dibuat dengan 2 sub-tipe node yaitu

- AtomNode yaitu node yang berdiri sendiri seperti G,J,K, dst
- dan ConnectorNode yaitu operator penghubung antara Atomnote seperti AND, OR,dst

Dimana setiap facts (fakta) diawali dengan tanda (=)
contoh :

=GJFK : bahwa node G,J,F,K bernilai true

Sedangkan query diawali dengan (?)
contoh :

?Y : untuk mengetahui kebenaran nilai dari Y

Alur kerja program :
1. node – node akan dilihat apakah memiliki turunan, apakah sudah pernah dikunjungi atau belom, serta melihat statenya apakah true atau false dalam class node

```python
class Node:
  def __init__(self, tree):
    self.children = [] # In A => B, => is child of B
    self.visited = False # When recursively parsing the Graph, it avoids infinite loop
    self.state = False # Saves if the result is True
```

2. Nama – nama node yang sudah diinputkan akan disimpan dalam class AtomNode

```
class AtomNode(Node):
  def __init__(self, name):
    super(AtomNode, self).__init__()
    self.name = name
```

3. Sedangkan tipe conector akan disimpan dalam class ConnectorNode, dalam class ini pula nilai operands akan disimpan yang nanti akan digunakan untuk menarik nilai kebenaran node conector itu sendiri.

```
class ConnectorNode(Node):
  def __init__(self, connector_type):
      super(ConnectorNode, self).__init__(tree)
      self.type = connector_type
      self.operands = [] # For example, in A + B, A and B are operands of +
      self.state = None
```

4. Aturan-aturan dan fakta-fakta yang diberikan akan dievaluasi dan hasil resolusi untuk setiap query disimpan dalam **results**.

```
def resolve_lines(parser):
    tree = Tree.NPITree(parser.structured_rules, parser.facts, parser.queries)
    results = {}
    for query in parser.queries:
        results[query] = tree.resolve_query(query)
        color = Color.GREEN if results[query] is True else Color.FAIL
        print(f"{ query } resolved as { color }{ results[query] }{ Color.END }")
    return results
```

5. Kemudian untuk mengetahui serta membaca argumen dari baris perintah, membaca aturan-aturan dari file input, memeriksa mode, mengeksekusi aturan secara lebih rinci dapat dilihat menggunkan mode interactive

```
if __name__ == "__main__":
    args = Cmd.args

    try:
        with open(args.input) as f:
            lines = f.readlines()

        if args.mode == "interactive":
            Prompt.ESPrompt(lines).cmdloop()
        else:
            parser = ESParser(lines)
            # ... (cek argumen dan eksekusi perintah berdasarkan argumen)
            if args.history:
                save_history(res)

    except (Exception, BaseException) as e:
        print(e)
        sys.exit(1)
```