

LAB Exam-2

Naman Agarwal
18m18ce057

variable = { 'P': 0, 'Q': 1, 'R': 2 }

priority = { '~': 3, '^': 2, 'v': 1 }

```
def eval (i, val1, val2)
```

```
    if i == '^':
```

```
        return val2 and val1
```

```
    return val2 or val1
```

```
def isOperand (c)
```

```
    return c.isalpha() and c != 'v'
```

```
def is hasLessOrEqualPriority (c1, c2):
```

```
    try:
```

```
        return Priority (c1) <= Priority [c2]
```

```
    except:
```

```
        return False.
```

```
def toPostfix (infix):
```

```
    stack = []
```

```
    postfix = ""
```

```
    for c in infix:
```

```
        if isOperand (c):
```

```
            postfix += c
```

```
        else
```

N. Agarwal

Naman Agarwal
18m18c2057

if $c == 'C'$

stack.append(c)

elif $c == '>'$

operator = stack.pop()

while not ~~is~~ operator != 'C'

postfix += operator

operator = stack.pop()

else :

while (not ~~stack~~ len-- != 0

and has less Or Equal Priority (c, peek(stack))

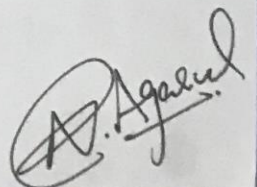
postfix += stack.pop()

stack.append(c)

while (stack != 0):

postfix += stack.pop()

return postfix.



Naman Agarwal
18m18CS057

```
def evaluate ( exp, comb )
```

```
    stack = []
```

```
    for i in exp:
```

```
        if isOperand(i):
```

```
            stack.append( comb[variable[i]] )
```

```
        elif i == 'or':
```

```
            val1 = stack.pop()
```

```
            stack.append( not val1 )
```

```
        else
```

```
            val1 = stack.pop()
```

```
            val2 = " ".pop()
```

```
            stack.append( eval(e, val1, val2) )
```

```
    return stack.pop()
```

```
def check()
```

```
    kb = (input("Enter knowledge base "))
```

```
    query = (input("Enter Query "))
```

```
    combinations = [ [True, True, True], [True, True, False],
```

```
                      [True, False, True], [True, False, False],
```

```
                      [False, True, True], [False, True, False],
```

N. Agarwal

Naman Agarwal
[False, True, True], [False, False, False]
18M18CS057

postfix_kb = toPostfix(kb)
postfix_q = toPostfix(q)

for combination in combinations :

eval_kb = evaluatePostfix(postfix_kb,
combination]

eval_q = evaluatePostfix(postfix_q,
combination]

print(combination, eval_kb, eval_q)

if (eval_kb == True):

if (eval_q == False):

return False

return True.

A. Agarwal