

Title: Implementation of Unification and Resolution

EX. NO : 07

Name: Parth Langalia

DATE : 14-03-2023

Reg No. : RA2011033010033

AIM :

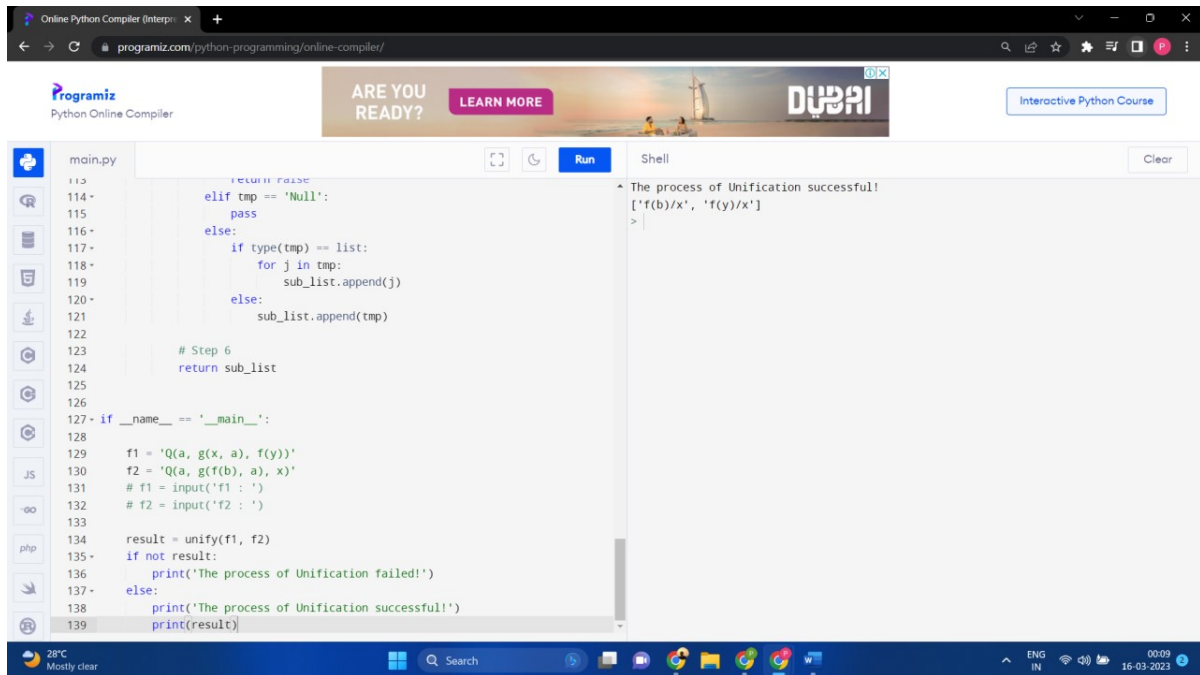
To Develop an optimized technique using an appropriate artificial intelligence algorithm to solve the Unification and Resolution

PSEUDO CODE :

1. function PL-RESOLUTION (KB, Q) returns true or false inputs: KB,
2. the knowledge base, group of sentences/facts in propositional logic
3. Q, the query, a sentence in propositional logic
4. clauses \rightarrow the set of clauses in the CNF representation of $KB \wedge Q$ new $\rightarrow \{ \}$
5. loop do for each C_i, C_j in clauses do
6. resolvents \rightarrow PL-RESOLVE (C_i, C_j)
7. if resolvents contains the empty clause the return true
8. new \rightarrow new union resolvents
9. if new is a subset of clauses then return false
10. clauses \rightarrow clauses union true

PROGRAM(with OUTPUT) :

Unification:



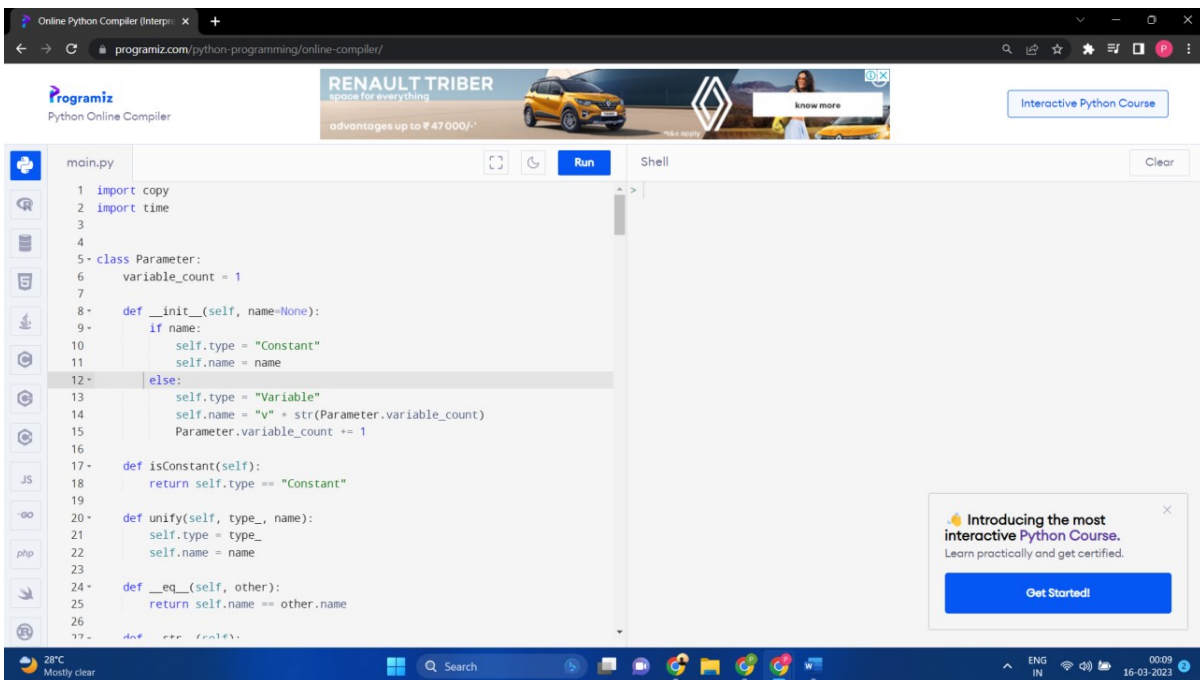
The screenshot shows the Programiz Online Python Compiler interface. The code editor contains a Python script for unification. The script defines a function `unify(f1, f2)` that checks if two expressions are unifiable. It uses a recursive approach, handling base cases like `Null` and lists. The main part of the script prompts the user for two expressions, `f1` and `f2`, and then calls `unify(f1, f2)`. The output in the shell shows that the process of unification was successful, returning the substitutions `{'f(b)/x', 'f(y)/x'}`.

```
main.py
113
114     return raise
115 elif tmp == 'Null':
116     pass
117 else:
118     if type(tmp) == list:
119         for j in tmp:
120             sub_list.append(j)
121     else:
122         sub_list.append(tmp)
123
124 # Step 6
125 return sub_list
126
127 if __name__ == '__main__':
128
129     f1 = 'Q(a, g(x, a), f(y))'
130     f2 = 'Q(a, g(f(b), a), x)'
131     # f1 = input('f1 : ')
132     # f2 = input('f2 : ')
133
134     result = unify(f1, f2)
135     if not result:
136         print('The process of Unification failed!')
137     else:
138         print('The process of Unification successful!')
139         print(result)
```

Shell

```
The process of Unification successful!
{'f(b)/x', 'f(y)/x'}
>
```

Resolution:



The screenshot shows the Programiz Online Python Compiler interface. The code editor contains a Python script for resolution. The script defines a class `Parameter` with attributes `variable_count` and `name`. It implements methods `__init__`, `isConstant`, `unify`, and `__eq__`. The main part of the script prompts the user for two expressions, `f1` and `f2`, and then calls `unify(f1, f2)`. The output in the shell shows that the process of resolution was successful, returning the substitutions `{'f(b)/x', 'f(y)/x'}`.

```
main.py
1 import copy
2 import time
3
4
5 class Parameter:
6     variable_count = 1
7
8     def __init__(self, name=None):
9         if name:
10             self.type = "Constant"
11             self.name = name
12         else:
13             self.type = "Variable"
14             self.name = "v" + str(Parameter.variable_count)
15             Parameter.variable_count += 1
16
17     def isConstant(self):
18         return self.type == "Constant"
19
20     def unify(self, type_, name):
21         self.type = type_
22         self.name = name
23
24     def __eq__(self, other):
25         return self.name == other.name
26
27     def __str__(self):
```

Shell

```
The process of Resolution successful!
{'f(b)/x', 'f(y)/x'}
>
```

Manual Calculations :

Manual Calculation / Input/output :

Unification :

$f_1: 'Q(a, g(n, a), f(y))'$

$f_2: 'Q(a, g(f(b), a), n)'$

Process of unification successful:

$['f(b)/x', 'f(y)/x']$

Resolution :

Input:

b

$A(Alice)$

$\cup A(Alice)$

$z(zig)$

$\cup z(zig)$

$g(golf)$

$\cup g(good)$

(0)

$A(u) \Rightarrow B(u)$

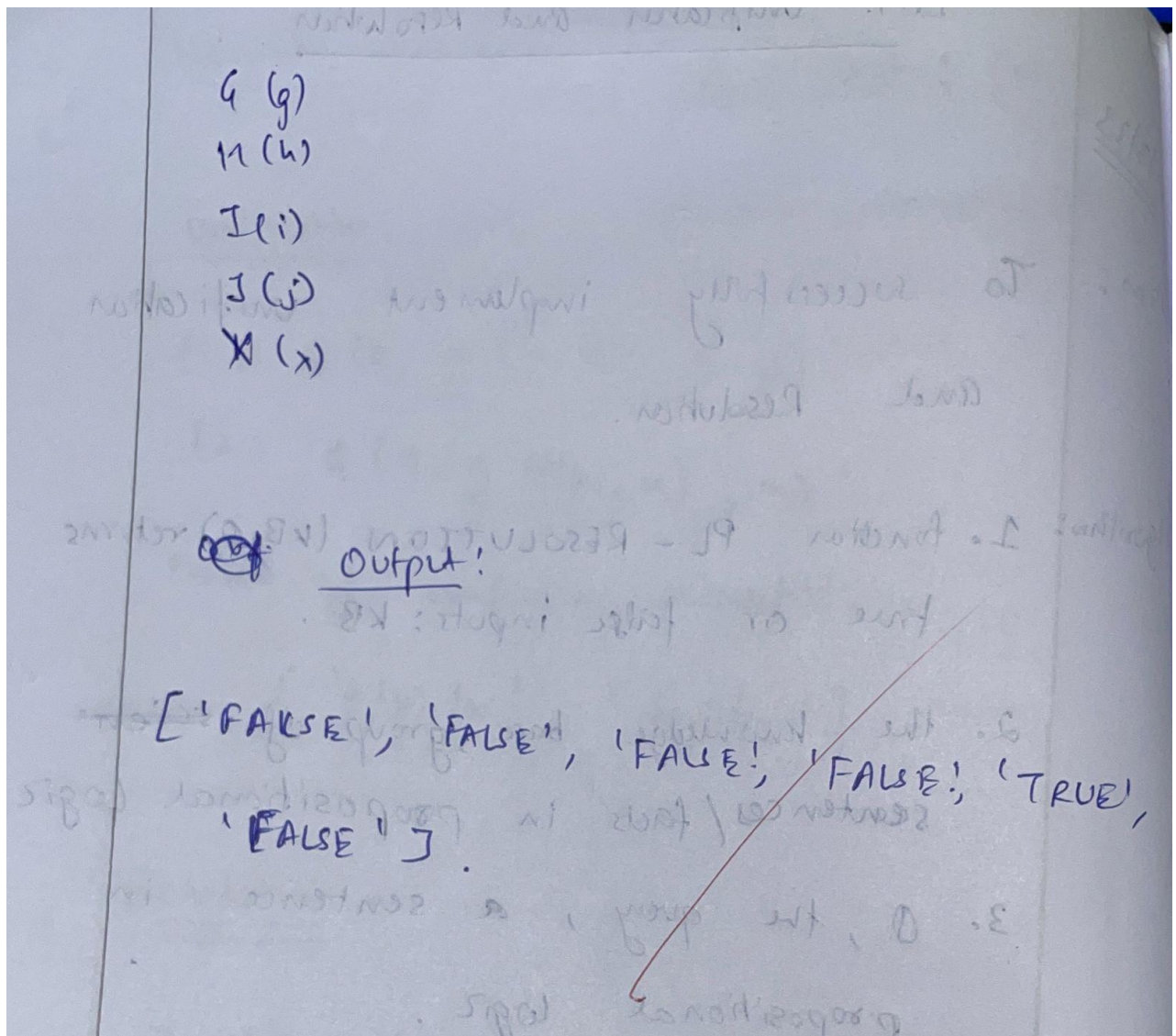
$B(u) \Rightarrow C(u)$

$C(u) \Rightarrow D(u)$

$D(u) \Rightarrow E(u)$

$E(u) \Rightarrow A(u)$





RESULT :

Developed Unification and Resolution Algorithm in Python for solving logical problems.