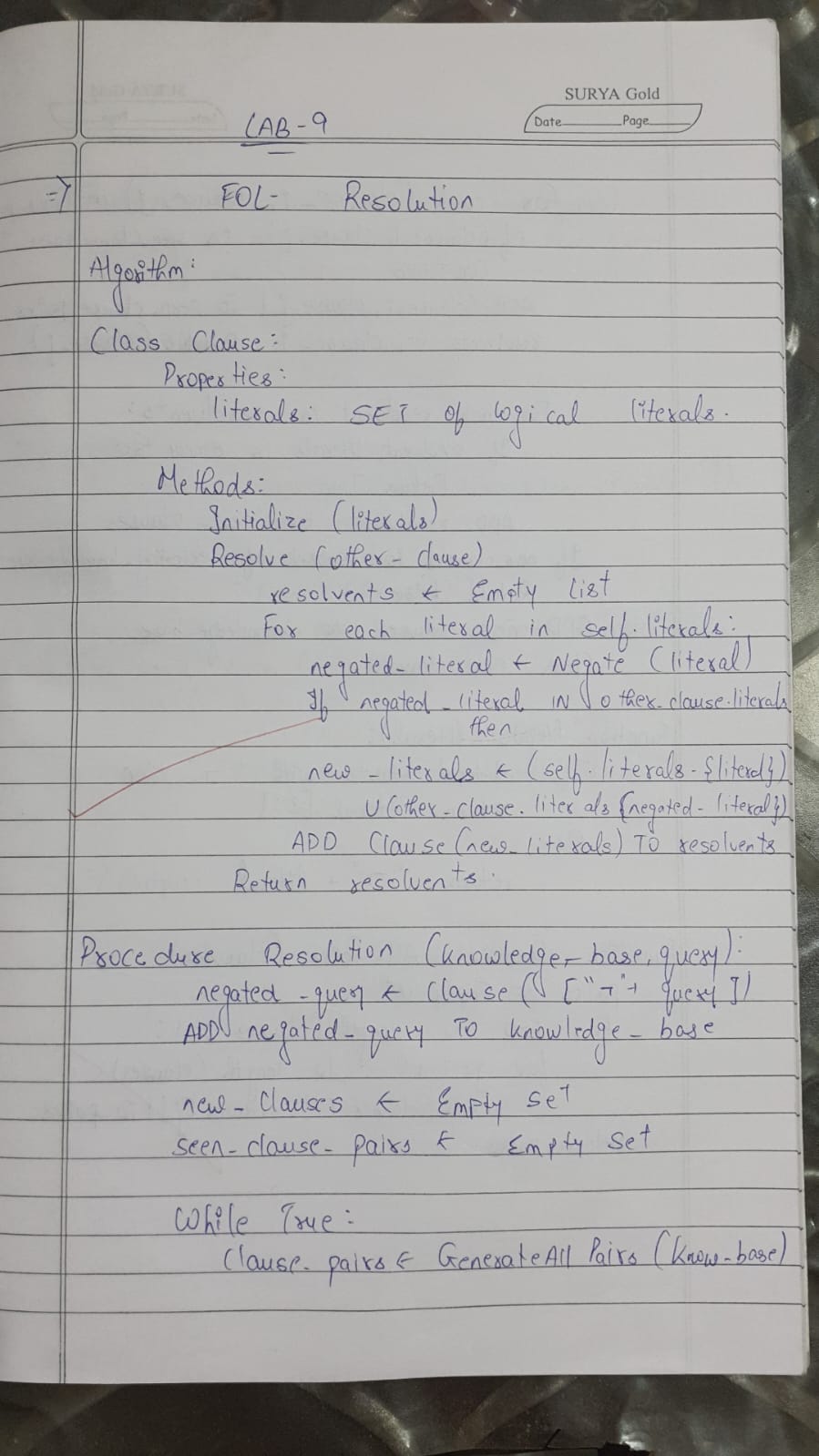
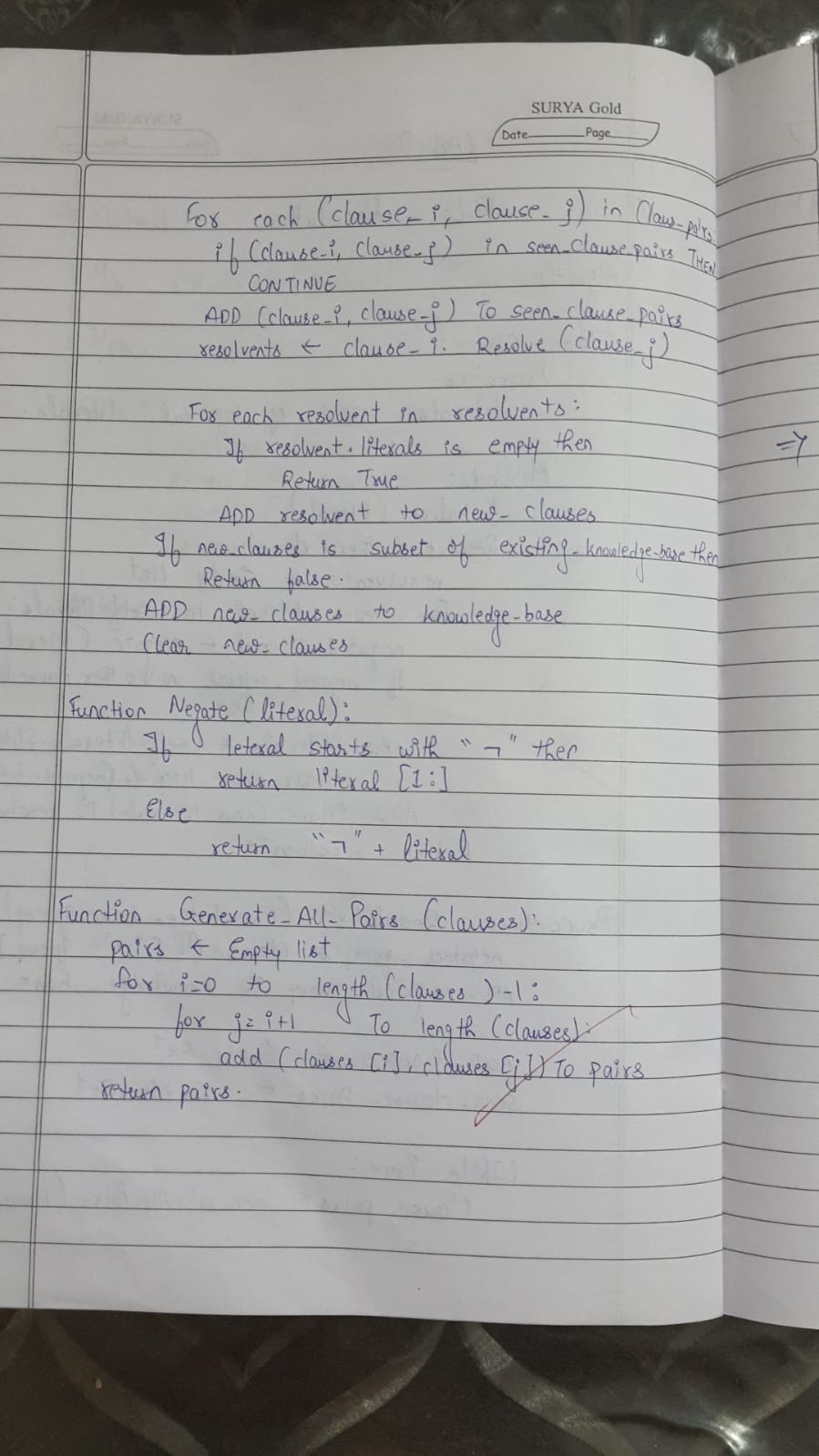
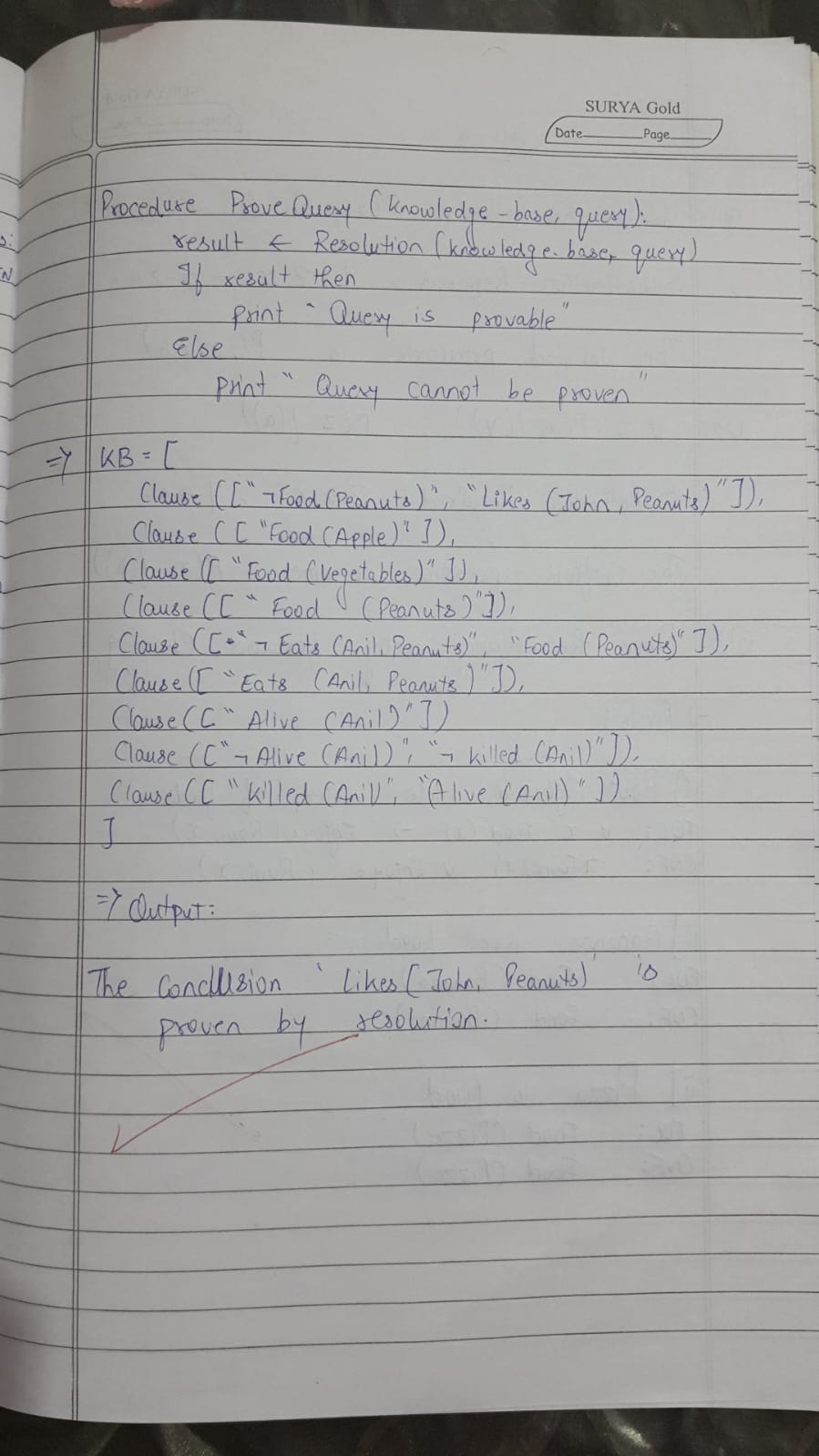
**First Order Logic – Resolution**

**Algorithm:**







**Code:**

class Clause:

    def \_\_init\_\_(self, literals):

        self.literals = set(literals)

    def \_\_repr\_\_(self):

        return " ∨ ".join(sorted(self.literals))

    def resolve(self, other):

        resolvents = []

        for literal in self.literals:

            negated\_literal = f"¬{literal}" if not literal.startswith('¬') else literal[1:]

            if negated\_literal in other.literals:

                new\_literals = (self.literals - {literal}) | (other.literals - {negated\_literal})

                resolvents.append(Clause(new\_literals))

        return resolvents

def resolution(clauses, query):

    negated\_query = Clause([f"¬{query}"])

    clauses.append(negated\_query)

    new = set()

    seen\_pairs = set()

    while True:

        pairs = [(clauses[i], clauses[j]) for i in range(len(clauses)) for j in range(i + 1, len(clauses))]

        for ci, cj in pairs:

            if (ci, cj) in seen\_pairs or (cj, ci) in seen\_pairs:

                continue

            seen\_pairs.add((ci, cj))

            resolvents = ci.resolve(cj)

            for resolvent in resolvents:

                if not resolvent.literals:

                    return True

                new.add(frozenset(resolvent.literals))

        if new.issubset(set(map(frozenset, (c.literals for c in clauses)))):

            return False

        clauses.extend(Clause(list(literals)) for literals in new - set(map(frozenset, (c.literals for c in clauses))))

        new.clear()

KB = [

    Clause(["¬Food(Peanuts)", "Likes(John, Peanuts)"]),

    Clause(["Food(Apple)"]),

    Clause(["Food(Vegetables)"]),

    Clause(["Food(Peanuts)"]),

    Clause(["¬Eats(Anil, Peanuts)", "Food(Peanuts)"]),

    Clause(["Eats(Anil, Peanuts)"]),

    Clause(["Alive(Anil)"]),

    Clause(["¬Alive(Anil)", "¬Killed(Anil)"]),

    Clause(["Killed(Anil)", "Alive(Anil)"])

]

query = "Likes(John, Peanuts)"

if resolution(KB, query):

    print(f"The conclusion '{query}' is proven by resolution.")

else:

    print(f"The conclusion '{query}' cannot be proven.")

**Output:**

