Tagle, Marc Neil V.
M002

Mr. Alexander D. Maralit II
CSDS101: Discrete Structures

FINAL EXAM AND OUTPUT

Instructions

Create a solution-oriented program that solves a real-world problem using a concept from Discrete Structures.

1. Select a Subtopic in Discrete Structures
2. Identify a Real-World Problem
3. Design Your Solution Program
4. Document Your Work

Subtopic Chosen

Relations and its Properties

Program Name

Relation Property Checker

Program Description

This program features **a terminal-based interface** that allows users to input any set of ordered pairs and check which **properties of relations** (reflexive, symmetric, transitive, and antisymmetric) hold. While it does not target a specific real-world problem, it is **designed to be flexible**, Users can input any elements that represent real-world items to form a set. This means the program can be used to **explore and analyze relationships in various contexts**, such as people, objects, or events. It serves as a **general-purpose tool** that helps apply the theoretical concept of relations to many possible real-life scenarios.

Summary of Program Features

- Terminal-based Interface
- Modular Design
- Object-Oriented Programming
- Flexible Input
- Educational Purpose
- Reusable Tool

Technical Specifications

- **Integrated Development Environments (IDE):** VS Code
- **Programming Language:** Python

Source Code

A. rproperty.py

```python
class Relation_Property_Checker: # IMPORTANTE: Gumagawa po ito ng isang class na tinawag na 'Relation Property Checker'.

    def __init__(self) -> None: # Ito po ang constructor ng class.
        pass # Wala pong attributes ang ating class.


    def show_title(self) -> None: # Gumagawa po ito ng isang function na tinawag na 'show_title'.
        print("--- RELATION PROPERTY CHECKER ---") # Pini-print po nito ang title ng program sa terminal.


    def get_set(self) -> list: # IMPORTANTE: Gumagawa po ito ng isang function na tinawag na 'get_set'.
        print("CREATE SET") # Pini-print po nito ang title ng menu ng paggawa ng set.
        print("Enter elements of your set.") # Pini-print po nito ang instructions sa paggawa ng set.
        print("Enter blank to proceed.\n") # Kapag tapos na po mag-enter ng elements, magp-press [Enter] nalang to proceed.

        s = [] # Ini-initialize po nito ang isang list para sa ating set.
        i = 1 # Ini-initialize po nito ang index para malaman kung pang-ilan na element na ang ine-enter.
        while True: # Gumagawa po ito ng isang while loop na umiikot habang hindi pa bini-break.
            element = input(f"Enter element {i}: ").strip() # Humihingi po ito ng element kay user kada iteration.

            if element == '': # Ni-che-check po nito kung ang ni-input ni user ay blank.
                break # Kapag blanko po ang ni-input ni user, ang while loop ay magb-break.

            if element not in s: # Ni-che-check po nito kung ang ni-input ni user ay wala sa list 's'.
                s.append(element) # Kapag wala po sa list 's' ang input, ito ay i-a-append sa list.
                i += 1 # Pagkatapos po i-append ang bagong element, i-i-increment ang index para sa kasunod na prompt.
            else: # Kapag hindi po nasatisfy ang kanyang if condition, ang sumusunod na code ang magru-run.
                print("Element already in set.") # Kapag ang ni-input po si user ay naka'y list 's' na, ito ang i-pi-print sa terminal.

        s.sort() # Ni-o-organize po nito si list 's' upang mas maging maayos tingnan 'pag ni-print ito sa terminal.
        return s # Nire-return po ng function na ito ang list 's'.


    def show_set(self, s) -> None: # Gumagawa po ito ng isang function na tinawag na 'show_set'.
        print(f"A = {{{', '.join(s)}}}") # Ni-pi-print po nito ang list 's' sa format na 'A = {a, b, c, ...}'.


    def get_relation(self, s) -> list: # IMPORTANTE: Gumagawa po ito ng isang function na tinawag na 'get_relation'.
        print("CREATE RELATION") # Pini-print po nito ang title ng menu ng paggawa ng relation.
        print("Enter ordered pairs of elements from your set in the format (a,b).") # Pini-print po nito ang instructions sa paggawa ng relation.
        print("Enter blank to proceed.\n") # Kapag tapos na po mag-enter ng pairs, magp-press [Enter] nalang to proceed.
        self.show_set(s) # Tinatawag po nito ang function na 'show_set' para ipakita sa user ang elements ng kanyang set.
        print() # Ito po ay for terminal-based UI purposes.

        r = [] # Ini-initialize po nito ang isang list para sa ating relation.
        i = 1 # Ini-initialize po nito ang index para malaman kung pang-ilan na pair na ang ine-enter.
        while True: # Gumagawa po ito ng isang while loop na umiikot habang hindi pa bini-break.
            pair = input(f"Enter pair {i}: ").strip() # Humihingi po ito ng isang pair kay user kada iteration.

            if pair == '': # Ni-che-check po nito kung ang ni-input ni user ay blank.
                break # Kapag blanko po ang ni-input ni user, ang while loop ay magb-break.

            if pair.startswith('(') and pair.endswith(')'): # Ni-che-check po nito kung nasa tamang format ang ni-enter ni user.
                elements = pair[1:-1].split(',') # Tinatanggal po nito ang parentheses para ma-access ang mismong a and b pair.
                if len(elements) == 2: # Ni-che-check po nito kung ang pair ay binary o may dalawang elements lamang.
                    a, b = elements[0].strip(), elements[1].strip() # Ni-e-extract po nito ang dalawang elements sa variables na a, b
                    if a in s and b in s: # Ni-che-check po nito kung si a at b ay elements ng set.
                        if (a,b) in r: # Ni-che-check po nito kung ang pair ay nasa list 'r' already.
                            print("Pair already in relation.") # Kapag nag-e-exist na po ang pair kay list 'r', ito po ang i-pi-print.
                        else: # Kapag hindi po nasatisfy ang kanyang if condition, ang sumusunod na code ang magru-run.
                            r.append((a, b)) # Kapag wala po sa list 'r' ang input, ito ay i-a-append sa list.
                            i += 1 # Pagkatapos po i-append ang bagong element, i-i-increment ang index para sa kasunod na prompt.
                    else: # Kapag hindi po nasatisfy ang kanyang if condition, ang sumusunod na code ang magru-run.
                        print("Element/s not in set.") # Kapag isa o parehas na elements ay wala sa set, ito po ang i-pi-print.
                else: # Kapag hindi po nasatisfy ang kanyang if condition, ang sumusunod na code ang magru-run.
                    print("Invalid pair format. Use (a,b).") # Kapag hindi po binary ang relation, ito po ang i-pi-print.
            else: # Kapag hindi po nasatisfy ang kanyang if condition, ang sumusunod na code ang magru-run.
                print("Invalid pair format. Use (a,b).") # Kapag hindi po naka-enclosed in parentheses ang pair, ito po ang i-pi-print.

        r.sort() # Ni-o-organize po nito si list 'r' upang mas maging maayos tingnan 'pag ni-print ito sa terminal.
        return r # Nire-return po ng function na ito ang list 'r'.


    def show_relation(self, r) -> None: # Gumagawa po ito ng isang function na tinawag na 'show_relation'.
        formatted_pairs = ', '.join(f'({a},{b})' for a, b in r) # Ni-co-comma separate po nito ang bawat pair kay list 'r'.
        print(f"R = {{{formatted_pairs}}}") # Ni-pi-print po nito ang list 'r' sa format na 'R = {(a,b), (c,d), ...}'.


    def is_reflexive(self, s, r) -> bool: # IMPORTANTE: Gumagawa po ito ng isang function na tinawag na 'is_reflexive'.
        for e in s: # Gumagawa po ito ng isang for loop na nag-i-iterate sa list 's' (ang ating set).
            if (e,e) not in r: # Ni-che-check po nito kung ang self-pair ng isang element ng set ay wala sa relation.
                return False # Kung wala po ang self-pair sa relation, i-re-return po ng function ay False.
        return True # Kapag lahat po ng self-pairs ng bawat set element ay nasa relation, i-re-return ng function ay True.


    def is_symmetric(self, r) -> bool: # IMPORTANTE: Gumagawa po ito ng isang function na tinawag na 'is_symmetric'.
        for a, b in r: # Gumagawa po ito ng isang for loop na nag-i-iterate sa list 'r' (ang ating relation).
            if (b, a) not in r: # Ni-che-check po nito kung ang symmetric-pair ng isang pair ay wala sa relation.
                return False # Kung wala pong symmetric pair ang isang pair sa relation, i-re-return po ng function ay False.
        return True # Kapag lahat po ng pair sa relation ay may symmetric pair, i-re-return ng function ay True.


    def is_antisymmetric(self, r) -> bool: # IMPORTANTE: Gumagawa po ito ng isang function na tinawag na 'is_antisymmetric'.
        for a, b in r:  # Gumagawa po ito ng isang for loop na nag-i-iterate sa list 'r' (ang ating relation).
            if (b, a) in r and a != b: # Ni-che-check po nito kung ang symmetric-pair ng isang pair ay nasa sa relation at ang magka-pair ay equal sa isa't isa.
                return False # Kapag may magka-symmetric po na pair sa relation at ang elements ng pairs ay hindi equal sa isa't isa, i-re-return ng function ay False.
        return True # Kapag wala pong nakitang nagsatisfy sa condition ng if block, i-re-return ng function ay True.


    def is_transitive(self, r) -> bool: # IMPORTANTE: Gumagawa po ito ng isang function na tinawag na 'is_transitive'.
        for a, b in r: # Kumukuha po ito ng bawat pares mula sa r at itatalaga sa a at b sa bawat ikot kay list 'r'.
            for c, d in r: # Kumukuha po ito ng bawat pares mula sa r at itatalaga naman sa c at d sa bawat ikot kay list 'r'.
                if b == c and (a, d) not in r: # Ni-che-check po nito kung may transitive chain ba na nasira.
                    return False # If me'ron pong transitive chain ang nasira o hindi kumpleto, i-re-return ng function ay False.
        return True # If wala pong transitive chain na hindi kumpleto, i-re-return ng function ay True.

    def show_properties(self, s, r) -> None: # Gumagawa po ito ng isang function na tinawag na 'show_properties'.
        print("RELATION PROPERTIES") # Pini-print po nito ang header bago ipakita ang properties ng relation.
        print(f"Reflexive: {'Yes' if self.is_reflexive(s, r) else 'No'}") # Pinapakita po nito kung reflexive ang relation o hindi.
        print(f"Symmetric: {'Yes' if self.is_symmetric(r) else 'No'}") # Pinapakita po nito kung symmetric ang relation o hindi.
        print(f"Antisymmetric: {'Yes' if self.is_antisymmetric(r) else 'No'}") # Pinapakita po nito kung antisymmetric ang relation o hindi.
        print(f"Transitive: {'Yes' if self.is_transitive(r) else 'No'}") # Pinapakita po nito kung transitive ang relation o hindi.
```

B. main.py

```python
from rproperty import Relation_Property_Checker # Ni-i-import po nito ang module/class na 'Relation_Property_Checker' sa file na 'rproperty'.
import os # Ni-i-import po nito ang 'os' module para sa purpose ng pagclear ng terminal.

def divider() -> None: # Gumagawa po ito ng isang function na tinawag na 'divider'.
    print("_" * 33, "\n") # Naqpi-print po ito ng divider na gawa sa mga underscores. Ito ay for terminal-based UI purposes.

def main() -> None: # Ginagawa po nito ang main function na magiging entry point ng user.
    rpc = Relation_Property_Checker() # Gumagawa po ito ng isang instance ng ating 'Relation_Property_Checker' class.

    while True: # Gumagawa po ito ng isang while loop na umiikot habang hindi pa bini-break.

        while True: # Gumagawa po ito ng isang while loop na umiikot habang hindi pa bini-break.
            rpc.show_title() # Tinatawag po nito ang function na 'show_title' para ipakita ang title ng program sa terminal.
            divider() # Tinatawag po nito ang function na 'divider' para magprint ng divider sa terminal.
            s = rpc.get_set() # Tinatawag po nito ang function na 'get_set' para gumawa si user ng set at ito ay itatalaga kay variable 's'.

            divider() # Tinatawag po nito ang function na 'divider' para magprint ng divider sa terminal.
            while True: # Gumagawa po ito ng isang while loop na umiikot habang hindi pa bini-break.
                proceed = input("Are you happy with your set? [y/n]: ") # Tinatanong po nito ang user kung gusto na niyang magproceed sa paggawa ng relation.
                if proceed == 'y': # Ni-che-check po nito kung ang response ni user ay 'y'.
                    os.system('cls') # Ni-c-clear po nito ang terminal.
                    break # Lalabas po sa inner loop gamit itong keyword.
                elif proceed == 'n': # Ni-che-check po nito kung ang response ni user ay 'n'.
                    os.system('cls') # Ni-c-clear po nito ang terminal.
                    break # Lalabas po sa inner loop gamit itong keyword.
                else: # Ni-che-check po nito kung ang response ni user ay hindi 'y' o 'n'.
                    print("Invalid response! Please enter [y/n].\n") # Ni-pi-print po ito kapag mali ang response ng user.

            if proceed == 'y': # Ni-che-check po nito kung ang response ni user ay 'y'.
                break # Lalabas po sa outer loop gamit itong keyword.

        while True: # Gumagawa po ito ng isang while loop na umiikot habang hindi pa bini-break.
            rpc.show_title() # Tinatawag po nito ang function na 'show_title' para ipakita ang title ng program sa terminal.
            divider() # Tinatawag po nito ang function na 'divider' para magprint ng divider sa terminal.
            r = rpc.get_relation(s) # Tinatawag po nito ang function na 'get_relation' para gumawa si user ng relation at ito ay itatalaga kay variable 'r'.

            divider() # Tinatawag po nito ang function na 'divider' para magprint ng divider sa terminal.
            while True: # Gumagawa po ito ng isang while loop na umiikot habang hindi pa bini-break.
                proceed = input("Are you happy with your relation? [y/n]: ") # Tinatanong po nito ang user kung gusto na niyang makita ang properties ng relation.
                if proceed == 'y': # Ni-che-check po nito kung ang response ni user ay 'y'.
                    os.system('cls') # Ni-c-clear po nito ang terminal.
                    break # Lalabas po sa inner loop gamit itong keyword.
                elif proceed == 'n': # Ni-che-check po nito kung ang response ni user ay 'n'.
                    os.system('cls') # Ni-c-clear po nito ang terminal.
                    break # Lalabas po sa inner loop gamit itong keyword.
                else: # Ni-che-check po nito kung ang response ni user ay hindi 'y' o 'n'.
                    print("Invalid response! Please enter [y/n].\n") # Ni-pi-print po ito kapag mali ang response ng user.

            if proceed == 'y': # Ni-che-check po nito kung ang response ni user ay 'y'.
                break # Lalabas po sa outer loop gamit itong keyword.

        rpc.show_title() # Tinatawag po nito ang function na 'show_title' para ipakita ang title ng program sa terminal.
        divider() # Tinatawag po nito ang function na 'divider' para magprint ng divider sa terminal.
        rpc.show_set(s) # Tinatawag po nito ang function na 'show_set' para ipakita ang nakaformat na set sa terminal.
        rpc.show_relation(r) # Tinatawag po nito ang function na 'show_relation' para ipakita ang nakaformat na relation sa terminal.
        print() # Ito po ay for terminal-based UI purposes.
        rpc.show_properties(s, r) # Tinatawag po nito ang function na 'show_properties' para ipakita ang mga properties ng relation.
        divider() # Tinatawag po nito ang function na 'divider' para magprint ng divider sa terminal.

        while True: # Gumagawa po ito ng isang while loop na umiikot habang hindi pa bini-break.
            another = input("Do you want to check the properties of another relation? [y/n]: ") # Tinatanong po nito ang user kung gusto niyang ulitin ang program.
            if another == 'y': # Ni-che-check po nito kung ang response ni user ay 'y'.
                os.system('cls') # Ni-c-clear po nito ang terminal.
                break # Lalabas po sa loop gamit itong keyword.
            elif another == 'n': # Ni-che-check po nito kung ang response ni user ay 'n'.
                print("\nProgram by M002 | Tagle, Marc Neil V.") # Ito po ay credits sa gumawa ng program.
                print("Thank you! Goodbye.") # Ito po ang parting message ng program.
                return # Kapag hindi na gusto ng user na magcheck ng isa pang relation, mag-re-return ang 'main' function at matatapos ang pagrun ng program.
            else: # Ni-che-check po nito kung ang response ni user ay hindi 'y' o 'n'.
                print("Invalid response! Please enter [y/n].\n") # Ni-pi-print po ito kapag mali ang response ng user.

if __name__ == '__main__': # Kapag ang pangalan po ng file ay 'main', magru-run ang sumusunod na function.
    main() # Tinatawag po nito ang 'main' function at magsisimula at program.
```

Output Screenshots

A. Run #1

```
--- RELATION PROPERTY CHECKER ---
--------------------------------

CREATE SET
Enter elements of your set.
Enter blank to proceed.

Enter element 1: Fred
Enter element 2: Jeff
Enter element 3: Alex
Enter element 4: Mica
Enter element 5:
--------------------------------

Are you happy with your set? [y/n]:
```

```
--- RELATION PROPERTY CHECKER ---
--------------------------------

CREATE RELATION
Enter ordered pairs of elements from your set in the format (a,b).
Enter blank to proceed.

A = {Alex, Fred, Jeff, Mica}

Enter pair 1: (Alex,Alex)
Enter pair 2: (Fred,Fred)
Enter pair 3: (Jeff,Jeff)
Enter pair 4: (Mica,Mica)
Enter pair 5: (Alex,Fred)
Enter pair 6: (Fred,Alex)
Enter pair 7: (Fred,Jeff)
Enter pair 8: (Alex,Jeff)
Enter pair 9:
--------------------------------

Are you happy with your relation? [y/n]:
```

```
--- RELATION PROPERTY CHECKER ---
--------------------------------

A = {Alex, Fred, Jeff, Mica}
R = {(Alex,Alex), (Alex,Fred), (Alex,Jeff), (Fred,Alex), (Fred,Fred), (Fred,Jeff), (Jeff,Jeff), (Mica,Mica)}

RELATION PROPERTIES
Reflexive: Yes
Symmetric: No
Antisymmetric: No
Transitive: Yes
--------------------------------

Do you want to check the properties of another relation? [y/n]: n

Program by M002 | Tagle, Marc Neil V.
Thank you! Goodbye.
```

B. Run #2 (Null Relation)

```
--- RELATION PROPERTY CHECKER ---
--------------------------------

CREATE SET
Enter elements of your set.
Enter blank to proceed.

Enter element 1: 1
Enter element 2: 2
Enter element 3: 3
Enter element 4: 4
Enter element 5: 5
Enter element 6:
--------------------------------

Are you happy with your set? [y/n]:
```

```
--- RELATION PROPERTY CHECKER ---
--------------------------------

CREATE RELATION
Enter ordered pairs of elements from your set in the format (a,b).
Enter blank to proceed.

A = {1, 2, 3, 4, 5}

Enter pair 1:
--------------------------------

Are you happy with your relation? [y/n]:
```

```
--- RELATION PROPERTY CHECKER ---
--------------------------------

A = {1, 2, 3, 4, 5}
R = {}

RELATION PROPERTIES
Reflexive: No
Symmetric: Yes
Antisymmetric: Yes
Transitive: Yes

--------------------------------

Do you want to check the properties of another relation? [y/n]: n

Program by M002 | Tagle, Marc Neil V.
Thank you! Goodbye.
```