

1. Draw the CFG (mark the nodes on the question paper), find the cyclomatic complexity using three formulae. Write three paths from the basic path set. [5+3+2]

def complex_function(matrix): ==> node 1 (start the node count from here)

result = [] → 2

for i in range(0, len(matrix), 1):

for j in range(0, len(matrix[i]), 1):

for k in range(0, len(matrix[i][j]), 1):

val = matrix[i][j][k] 12

if val % 2 == 0: 13

if val > 50: 14

if val % 5 == 0: 15

if val < 100: 16

result.append("Special Even") 17

else:

result.append("Large Even") 18

else:

result.append("Even > 50") 19

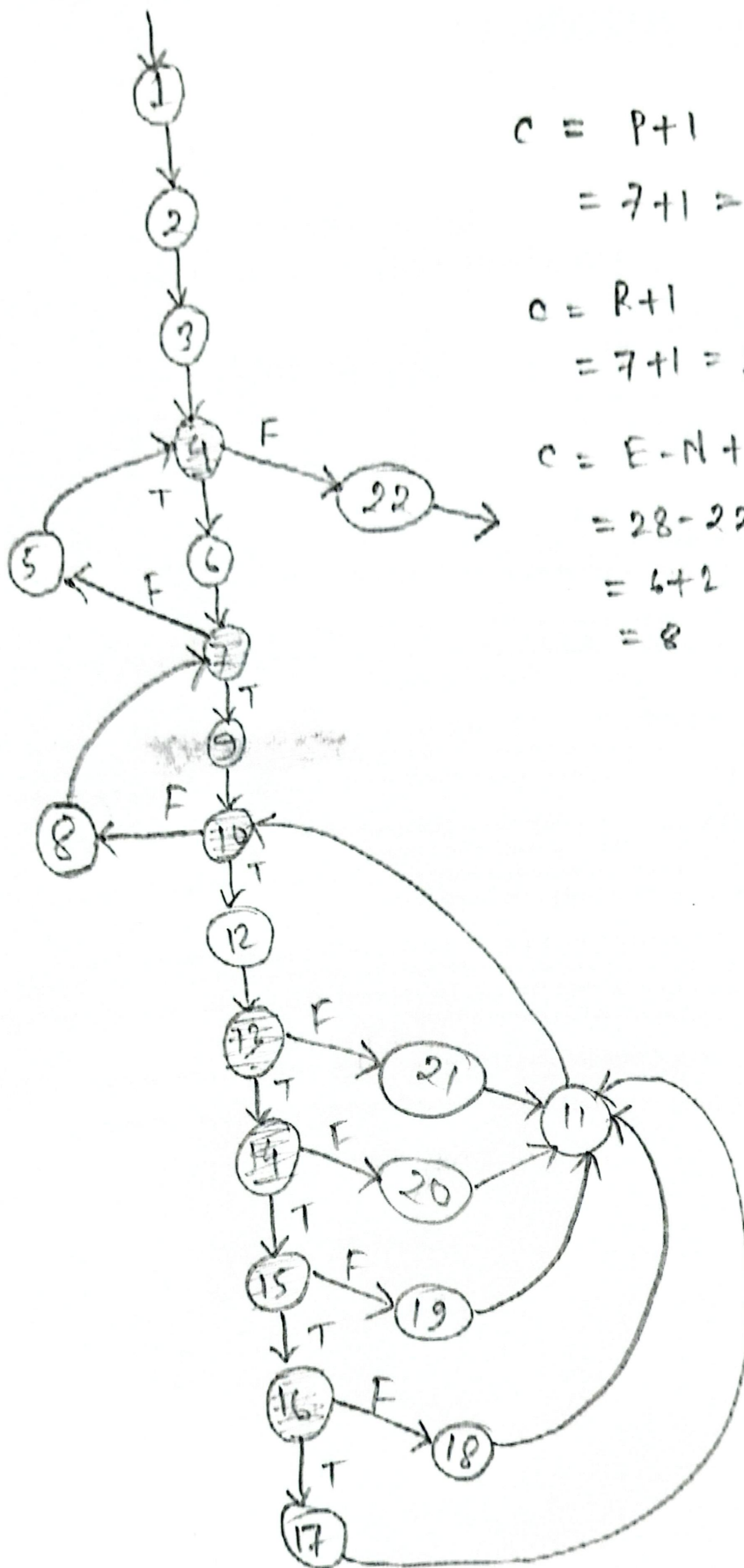
else:

result.append("Small Even") 20

else:

result.append("Odd") 21

return result 22



$$C = P + 1$$

$$= 7 + 1 = 8$$

$$C = R + 1$$

$$= 7 + 1 = 8$$

$$C = E - N + 2P$$

$$= 28 - 22 + 2$$

$$= 6 + 2$$

$$= 8$$

1. Draw the CFG (mark the nodes on the question paper), find the cyclomatic complexity using three formulae. Write **three paths** from the basic path set. [5+3+2]

```
def cooling_system_simulation(): # this is -- node 2
    temperature = 50
    max_temperature = 80
    fan_speed = 5
    cooling_rate = 2
    is_cooling_system_on = False
    is_emergency_mode = False

    while temperature > 0:
        if is_cooling_system_on:
            for i in range(0, fan_speed, 1):
                if temperature < max_temperature:
                    for j in range(0, cooling_rate, 1):
                        temperature -= 1
                        print(f"Cooling... Current temperature: {temperature}")
                    else:
                        print("Temperature is too high!")
                else:
                    is_cooling_system_on = False
                    temperature += 1
                    print(f"Warming... Current temperature: {temperature}")
            if is_emergency_mode:
                temperature -= 5
                return "Emergency mode activated! Forcing rapid cooling."
            if temperature <= 0:
                print("System shutting down. Temperature is safe.")
                break

    if __name__ == "__main__":
        print(cooling_system_simulation()) # start from here -- node 1
```

Handwritten annotations in red ink indicate node numbers for Control Flow Graph (CFG) construction:

- Node 2: Start of the function definition.
- Node 3: End of the initialization block (temperature, max_temperature, fan_speed, cooling_rate, is_cooling_system_on, is_emergency_mode).
- Node 4: Start of the while loop (while temperature > 0).
- Node 5: Start of the if is_cooling_system_on block.
- Node 6: Start of the for i in range(0, fan_speed, 1) loop.
- Node 7: Start of the if temperature < max_temperature block.
- Node 8: Start of the for j in range(0, cooling_rate, 1) loop.
- Node 9: Start of the temperature -= 1 statement.
- Node 10: Start of the print(f"Cooling... Current temperature: {temperature}") statement.
- Node 11: End of the for j in range(0, cooling_rate, 1) loop.
- Node 12: Start of the else block (print("Temperature is too high!")).
- Node 13: End of the if is_cooling_system_on block.
- Node 14: Start of the if is_emergency_mode block.
- Node 15: Start of the if temperature <= 0 block.
- Node 16: Start of the temperature -= 5 statement.
- Node 17: Start of the return "Emergency mode activated! Forcing rapid cooling." statement.
- Node 18: End of the if is_emergency_mode block.
- Node 19: End of the while loop.
- Node 20: Start of the if __name__ == "__main__": block.
- Node 21: End of the print(cooling_system_simulation()) statement.

$$C = P + 1 = 7 + 1 = 8$$

$$C = R + 1 = 7 + 1 = 8$$

$$C = E - N + 2P = 28 - 22 + 2 \times 1 = 8$$

