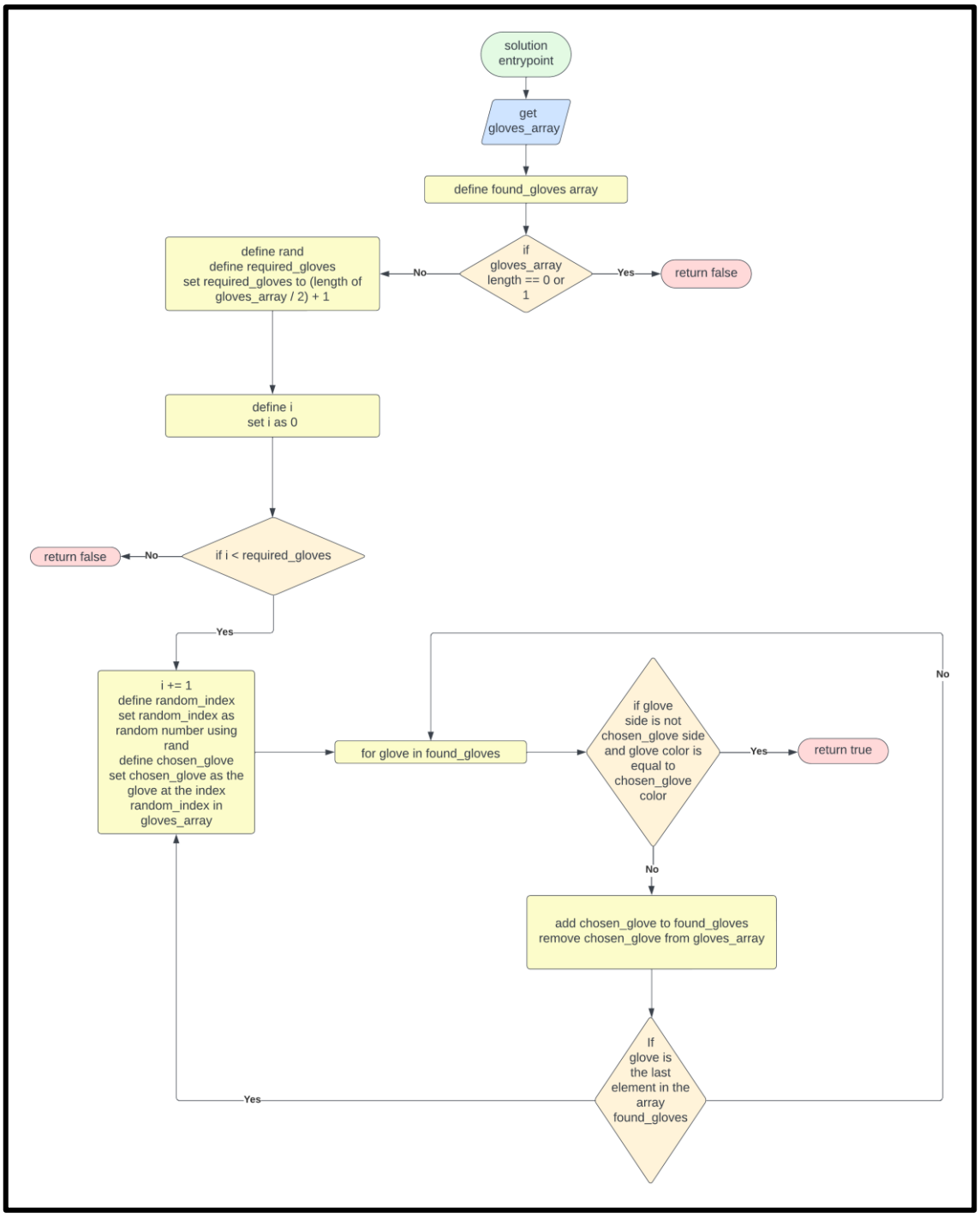
Pseudocode

```
BEGIN
instantiate gloves_array, found_gloves
FOR i IN range(5)
ADD Glove('r','black') TO gloves_array
                ADD Glove('l','black') TO gloves_array
END FOR

FOR i in range(3)
                ADD Glove('r','brown') TO gloves_array
                ADD Glove('l','brown') TO gloves_array
END FOR

FOR i in range(2)
                ADD Glove('r','brown') TO gloves_array
                ADD Glove('l','brown') TO gloves_array
END FOR

rand := random()
required_gloves := int((len(gloves_array)/ 2) + 1)
found_pair := false

FOR glove_index IN range(required_gloves)
random_index := rand.randint(0,len(gloves_array_ - 1)
chosen_glove := gloves_array[random_index]
END FOR

FOR compare_glove IN found_gloves
IF NOT chosen_glove.side == compare_glove.side
                AND chosen_glove.color == chosen_gloven.color THEN
found_pair := true
END IF NOT
END FOR

ADD (chosen_glove) TO found_gloves
REMOVE (chosen_glove) TO gloves_array
RETURN found_pair

located := true

FOR i IN range(1_000_000)
IF NOT find_glove() THEN
located := false
END IF NOT

IF NOT located THEN
PRINT ("Glove was unable to be found for some reason.")
ELSE PRINT ("A pair of gloves was found on every execution.")
END IF NOT
END FOR
END
```

Flowchart



Computational Complexity

```
from random import Random
from dataclasses import dataclass


@dataclass
class Glove:
    side: str                                                                   O(1)
    color: str                                                                  O(1)


def find_glove():
    # instantiate array
    gloves_array = []                                                           O(1)
    found_gloves = []                                                           O(1)

    # fill the gloves array with gloves. (5 pairs black, 3 pairs brown, 2 pairs grey)
    for i in range(5):                                                          O(n)
        gloves_array.append(Glove('r', 'black'))                                O(n)
        gloves_array.append(Glove('l', 'black'))                                O(n)
    for i in range(3):                                                          O(n)
        gloves_array.append(Glove('r', 'brown'))                                O(n)
        gloves_array.append(Glove('l', 'brown'))                                O(n)
    for i in range(2):                                                          O(n)
        gloves_array.append(Glove('r', 'grey'))                                 O(n)
        gloves_array.append(Glove('l', 'grey'))                                 O(n)

    rand = Random()                                                             O(1)
    required_gloves = int((len(gloves_array) / 2) + 1)                          O(1)
    found_pair = False                                                          O(1)

    for glove_index in range(required_gloves):                                  O(n)
        random_index = rand.randint(0, len(gloves_array) - 1)                   O(n)
        chosen_glove = gloves_array[random_index]                               O(n)

        for compare_glove in found_gloves:                                      O(n^2)
            if not chosen_glove.side == compare_glove.side and                  O(n^2)
chosen_glove.color == chosen_glove.color:                                       O(1)
                found_pair = True                                               O(1)

        found_gloves.append(chosen_glove)                                       O(1)
```

```
        gloves_array.remove(chosen_glove)                    O(1)

    return found_pair                                        O(1)


if __name__ == '__main__':                                   O(1)
    located = True                                           O(1)

    for i in range(1_000_000):                               O(n)
        if not find_glove():                                 O(1)
            located = False                                  O(1)

    if not located:                                          O(n)
        print("Glove was unable to be found for some reason.")  O(1)
    else:                                                    O(n)
        print("A pair of gloves was found on every execution.")  O(1)
```

O(1) = constant
O(n) = Linear
O(n^2) = Quadratic

Overall, the time complexity is O(n^2) which is a quadratic. This is because in the worst case, every iteration iterates over a subset of the array gloves_array which has a maximum number of elements n. So therefore logically, the larger n is, the more iterations over that subset array are required. Even though the n is divided by 2, the second iteration of all the gloves inside of found_gloves still iterates over an array with a length relative to n. So, the expanded complexity is O(n*[some multiple] * n*[some multiple]) since the static multiples of n are typically ignored if they're static numbers, the complexity is O(n*n) or O(n^2)

Each iteration
O(1) * 17 = O(17)
O(n)* 16 = O(16n)
O(n^2)* 2

ChatGPT Code & Comparison

Generated Code:

```
def min_gloves_to_select():
    # Define the number of pairs for each color
    pairs = {'black': 5, 'brown': 3, 'grey': 2}

    # Initialize the count for each color
    count = {'black': 0, 'brown': 0, 'grey': 0}

    # Initialize the total count of gloves selected
    total_count = 0

    # Iterate until we have at least one matching pair of each color
    while any(count[color] < pairs[color] for color in pairs):
        # Select a glove randomly
        selected_color = select_glove()

        # Update the count for the selected color
        count[selected_color] += 1

        # Increment the total count of gloves selected
        total_count += 1

    return total_count

def select_glove():
    # This function simulates selecting a glove randomly
    import random

    colors = ['black', 'brown', 'grey']
    return random.choice(colors)

# Run the simulation
result = min_gloves_to_select()
print(f"The smallest number of gloves to select is: {result}")
```

Comparison:

The primary difference between the ChatGPT generated implementation and our implementation is the inclusion of handedness to the gloves. Not all gloves are reversible, so implicitly stating the hands might be necessary depending on the type of glove. Another difference is how the return types are implemented. In the ChatGPT code, the solution returns the minimum number of gloves required to find an absolute match, in our code this is only a section of the solution: Ours is implemented in a test-like Boolean function which determines whether a matching glove pair has been found with the return value.

Team Evaluation

Pseudocode – Rahat Nafees 23015702
Flowchart – Tayyib Khan 21007967
Computational complexity – Sara Ali 22010736, Saadia Ali 23010730
Implementation – Charlie Hepworth 23021790

All members of this group equally contributed by completing the tasks they were given.