

TEAM 4A :

Juwairia Murtaza UB: 22007271

Aadam Ahmed UB: 23018946

Rayees Khan UB: 23033683

Aziz Al-Ali UB: 23017006

We were assigned this problem below:

Problem 2: The first 50 numbers – 1,2,...,50 – are written on a board. You have to apply the following operation 49 times; select two of the number on the board, a and b, write the absolute value of their difference $|a - b|$ on the board, and then erase both a and b. Write an algorithm that determines the remaining number that can be obtained in this manner.

PSEUDO CODE:

Get user input on desired range of array

Assign lowerbound the value of 1

Assign upperbound the value of the users input

Create and empty array called myList

While the length of myList is less than the upperbound:

 rangeOfList: Fill myList with numbers at random between the lower and upper bound

 if rangeOfList is not in myList:

 add it to the end of the array (myList)

sort the array myList so it is an ordered list (starting from 1)

print "Your array:", myList

create a counter and assign it the value of 0

while counter is less than upperbound - 1:

 pick an element at random from the array (myList)

 assign this element the value 'a' and remove it from the array: myList

 pick another element at random from the array (myList)

 assign this element the value 'b' and remove it from the array: myList

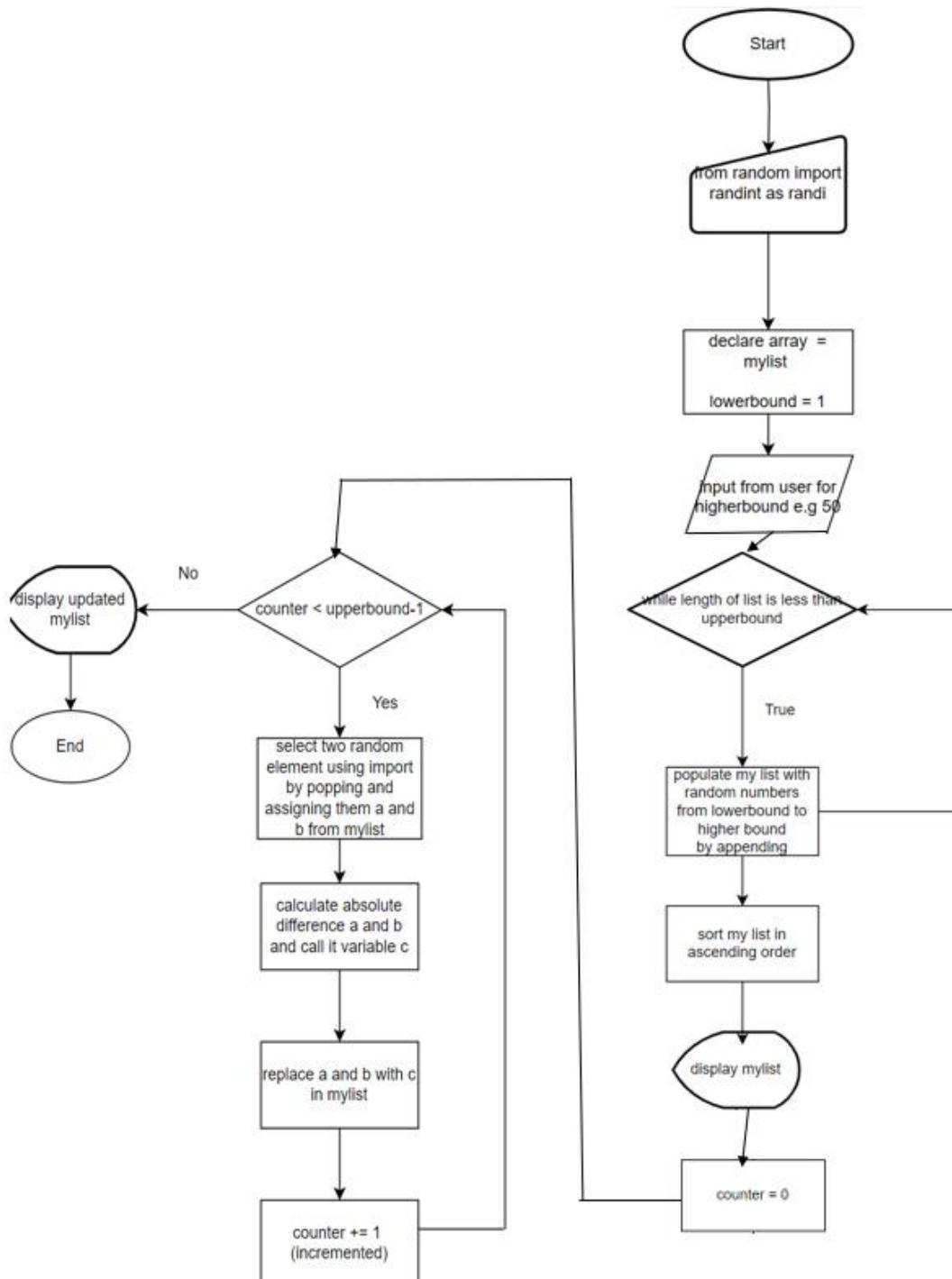
 calculate the absolute difference between 'a' and 'b' and assign it the value 'c'.

 add c to myList

 increment counter by 1

print "The final number left in your array is:", myList

FLOWCHART:



COMPARISON

AI generated code:

```
def remaining_number():
    # Initialize the list with the first 50 numbers
    numbers = list(range(1, 51))

    # Perform the operation 49 times
    for _ in range(49):
        # Select two numbers from the list
        a = min(numbers)
        numbers.remove(a)
        b = min(numbers)
        numbers.remove(b)

        # Calculate the absolute difference and add it to the list
        diff = abs(a - b)
        numbers.append(diff)

    # The remaining number is the only one left in the list
    return numbers[0]

# Call the function and print the result
result = remaining_number()
print("Remaining number:", result)
```

Our code:

```
from random import randint as randi

# The user input decides the size of the array
lowerbound = 1
upperbound = int(input("Enter the range of your array: "))

# myList is an array that formed form the data given by the user

myList = []
while len(myList) < upperbound:
    rangeOfList = randi(lowerbound,upperbound)
    if rangeOfList not in myList:
        myList.append(rangeOfList)
myList.sort()
print ("Your array:", myList)
# a and b are 2 elements/numbers chosen at random from the array "myList"
# c is the absolute difference between a and b in each calculation that is done
counter = 0
while counter < (upperbound-1):
    firstElement = randi(0, len(myList) - 1)
    a = myList.pop(firstElement)

    secondElement = randi(0, len(myList) - 1 )
    b = myList.pop(secondElement)

    c = abs(a - b)
    myList.append(c)
    counter += 1
print ("The final number left in your array is:",myList)
```

There is a critical difference between the code we developed and the AI-generated code. Our approach starts with an input line that prompts the user to input the range of their array, which is then used to create an array and perform calculations accordingly. In contrast, the AI-generated code initialises an array with values ranging from 1 to 50 and performs calculations based on these values. Our approach is more dynamic and flexible, allowing users to customise their array instead of the 1-50 array in the AI-generated code, which is rigid and not customisable.

Another significant difference between the two codes is the manner in which the loop is repeated. Our code uses a general statement to determine the number of times the loop should be repeated, which is one time less than the upper bound value. Conversely, the AI-generated code specifies that the loop should be repeated 49 times, which is less flexible. Additionally, we have employed the `sort()` function to ensure that the array is printed in an ordered list.

The AI-generated code assigns the variable '`a`' to the smallest number in the array using the `min()` function. In contrast, we have utilised `randi` from the `random` function to randomly choose an element from the array and assign it as '`a`'. This approach is more suitable for the problem we were assigned since it ensures that the value remaining at the end of the program will differ every time it is run. If we used the AI-generated approach, the value left at the end would be the same every time the program is run.

To ensure that '`a`' and '`b`' are not assigned the same element, we have used a single line of code to assign '`a`' a value and remove it from the list using the `pop()` function. This mechanism guarantees that '`a`' and '`b`' are always assigned different values. The AI-generated code, however, uses the `remove()` function to remove '`a`' and '`b`' from the array.

Both our code and the AI-generated code use the `abs()` function to calculate the difference between '`a`' and '`b`' and the `append()` function to add the calculation result to the end of the array and print the remaining value. However, our code is more dynamic and flexible, making it more suitable for use in an industry setting than the AI-generated code.

SAMPLES OF THE CODE RUNNING:

```
Enter the range of your array: 50
Your array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50]
The final number left in your array is: [7]
```

```
Enter the range of your array: 12
Your array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
The final number left in your array is: [2]
```

```
Enter the range of your array: 124
Your array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63,
64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 1
22, 123, 124]
The final number left in your array is: [84]
```

COMPUTATIONAL COMPLEXITY

```
from random import randint as randi

lowerbound = 1                                O (1)
rangeOfArray = int(input("Enter the range of your array: "))      O(1)

myList = []                                     O(1)

while len(myList) < upperbound:                O(n)
    rangeOfList = randi (lowerbound,upperbound + 1)      O(1)
    if rangeOfList not in myList:                      O(n)
        myList.append(rangeOfList)                     O (1)
    myList.sort()                                    O(n log n)

print ("Your array:" ,myList)

counter = 0                                     O (1)
while counter < (upperbound-1):                  O(n)
    firstElement = randi (0, len (myList) - 1)      O(1)
    a = myList[firstElement]                        O (1)
    myList.pop(firstElement)                       O (n)
    secondElement = randi (0, len (myList) -1 )     O(1)
    b = myList[secondElement]                      O (1)
    myList.pop(secondElement)                      O (n)
    c = abs(a - b)                               O (1)
    myList.append(c)                             O (1)
    counter += 1                                 O (1)

print ("The final number left in your array is:" , myList)
```

Summary:

The overall complexity for our code is $O(n \log n)$. This complexity shows that $\log n$ operations will occur n times on this line of code. The line that is being referred to here is `myList.sort()`. This overall complexity shows our programme is efficient and will perform well when a large range is entered by the user. This complexity usually suggests that the programmes time increases as the input increases so in the context of our code the bigger the array the longer the programme will take to run.(An array of 100 will have a longer running time than an array of 10.)

EVALUATION

Juwairia Murtaza: 10/10

Juwairia has been an invaluable team member throughout the project. She has assisted with creating the pseudo-code, flowchart, and complexity and led the actual code development. She has maintained open communication with her teammates and provided assistance whenever needed.

Adam Ahmed: 10/10

Adam has played a crucial role in the project. He has contributed to the actual code and pseudo-codes and led the flowchart. He has also significantly contributed to the complexity of the code. Adam has been an effective communicator, constantly engaging with and assisting his team members.

Aziz Al-Ali: 10/10

Aziz has demonstrated a strong work ethic and prioritised the development of the pseudo-code. He has worked collaboratively with his teammates, improving his pseudo-code, and communicating effectively with them.

Rayees Khan: 10/10

Rayees contributed to improving the code and making it more dynamic. He communicated efficiently with his teammates throughout the project and readily helped anyone who needed it. He also contributed to the computational complexity of the code.