



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Student Number : 2420133 Course Number : ACHA4020 Lecturer Name : Dr Seena Joseph
--

1.a The purpose of this flowchart is to find and display all prime numbers between a given start (Num1) and end number; [Table 1.0 shows a breakdown of the flowchart]

1.	Start
2.	Input start and end values
3.	Set Num1= Start
4.	Loop from Num1 to end; Check; if Num1 < End → Continue Check if Num1 > 1 (0 and 1 aren't prime) • isCorrect = True • Start checking factors using Num2 = 2
5.	Inner Loop to check if Num1 is prime: • While Num2 < Num1: ○ Check if Num1 % Num2 == 0 ■ If Yes → Not prime → isCorrect = false ○ Else: increment Num2
6.	After checking all factors: • If isCorrect == true ○ Add Num1 to list of primes
7.	Move to next Num1 (increment)
8.	Repeat until Num1 reaches End
9.	Print all prime numbers found
10.	Stop

1b.

There are 2 loops in the flowchart and both have important but different roles.

Loop 1: Outer Loop (over Num1)

Role:

- This loop goes through each number between Start and End (Num1).
- It checks each one individually to see if it's a prime.
- After each check, it increments Num1 and repeats until it hits the End value.

This is the main loop responsible for scanning the full number range.

Loop 2: Inner Loop (over Num2)

Role:

- For every number (Num1), this second loop tests if it's divisible by any number from 2 up to Num1-1.
- That's how it checks if Num1 is a prime.
- If it finds a factor (meaning Num1 is divisible by Num2), it sets `isCorrect = false` (not a prime).
- If no factors are found, Num1 is considered a prime.

2a

The inner loop is used to check if a number is prime. It keeps checking if the number can be divided by smaller numbers. If it finds one that works, it means the number is not prime and it stops early. This helps the program move on to the next number faster, so it doesn't waste time checking more numbers that it doesn't need to.

2b

The Boolean variable is called `isCorrect`. It is used to keep track of whether the number is prime or not. At the start of checking a number, it is set to `true`. If the number can be divided by another number, `isCorrect` is changed to `false`. At the end, if `isCorrect` is still `true`, the number is added to the list of primes.

3a

A conditional statement is used in programming to check if something is true or false. It helps the program decide what to do next based on the condition. If the condition is true, it runs one part of the code. If not, it can skip it or do something else.

3b

If Num1 > 1

This checks that the number is more than 1 because 0 and 1 are not prime. So the program ignores those.

If Num1 % Num2 == 0

This checks if Num1 can be divided by Num2 with no remainder. If it can, then Num1 is not a prime number.

If isCorrect == True

This checks if the number passed all the tests. If it did, the number is added to the list. If not, it gets skipped.

4a

```
def generate_special_numbers(start, end):  
    values = []  
    for num1 in range(start, end + 1):  
        if num1 > 1:  
            isCorrect = True  
            num2 = 2  
            while num2 < num1:  
                if num1 % num2 == 0:  
                    isCorrect = False  
                    break  
                num2 += 1  
            if isCorrect == True:  
                values.append(num1)  
    print("Prime numbers between", start, "and", end, "are:", values)
```

4b

```
generate_special_numbers(3, 16)
```

Output: Prime numbers between 3 and 16 are: [3, 5, 7, 11, 13]

```
generate_special_numbers(25, 42)
```

Output: Prime numbers between 25 and 42 are: [29, 31, 37, 41]

5a

Prime numbers between 12 and 22 are: [13, 17, 19]

The program goes through all the numbers in that range and checks if they are prime. It adds the ones that pass.

5b

Start is bigger than end. The program doesn't crash but the range is empty, so no numbers are checked and it prints an empty list.

5c

Start is a negative number. The program still works and just ignores the numbers that are 0 or less. It only checks from 2 upwards. Output: [2, 3, 5, 7]

5d

None of the numbers between 24 and 27 are prime. The program checks each one and finds no primes, so it gives back an empty list.

5e

Start and end are the same. The program checks that one number. Since 13 is a prime number, it gets added to the list. Output: [13]

6

The program uses a simple optimization. If it finds one number that divides evenly into the number being checked, it stops straight away. This saves time, especially for bigger numbers. It avoids checking more values once it knows the number isn't prime. This makes the program faster and helps with time complexity.

7

One of the hardest things for me was understanding how loops and conditionals work together. At first it was confusing, especially when you have a loop inside another loop. I overcame it by practising small problems and using flowcharts to see how the program flows. I also tried writing the code out slowly and using print statements to understand what's happening. In this assignment, I used loops to go through numbers and conditionals to check for prime numbers. That helped me get better with logic and problem solving.