21BDS0340

Abhinav Dinesh Srivatsa

BCSE101E, VL2021220107100 – TC2 (Theory)

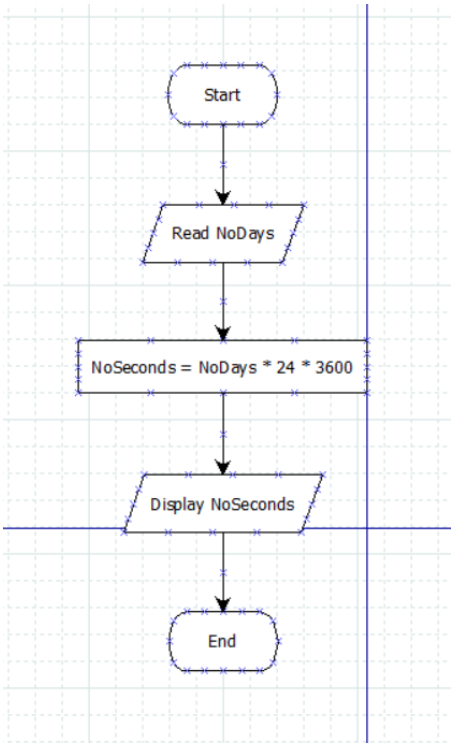
BCSE101E, VL2021220107101 – L15+L16+L29+L30 (Lab)

M12_CSQ1: Write a problem analysis chart (PAC), Algorithm and flowchart to calculate the age of a housefly in seconds, given the number of days the housefly lived.

PAC:

Data	Processing	Output	Alternative Solutions
NoDays = user input	NoSeconds = NoDays * 24 * 3600	NoSeconds	Define NoDays

Flowchart:



Algorithm:

Read NoDays

Calculate NoSeconds = NoDays * 24 * 3600

Display NoSeconds

Code:

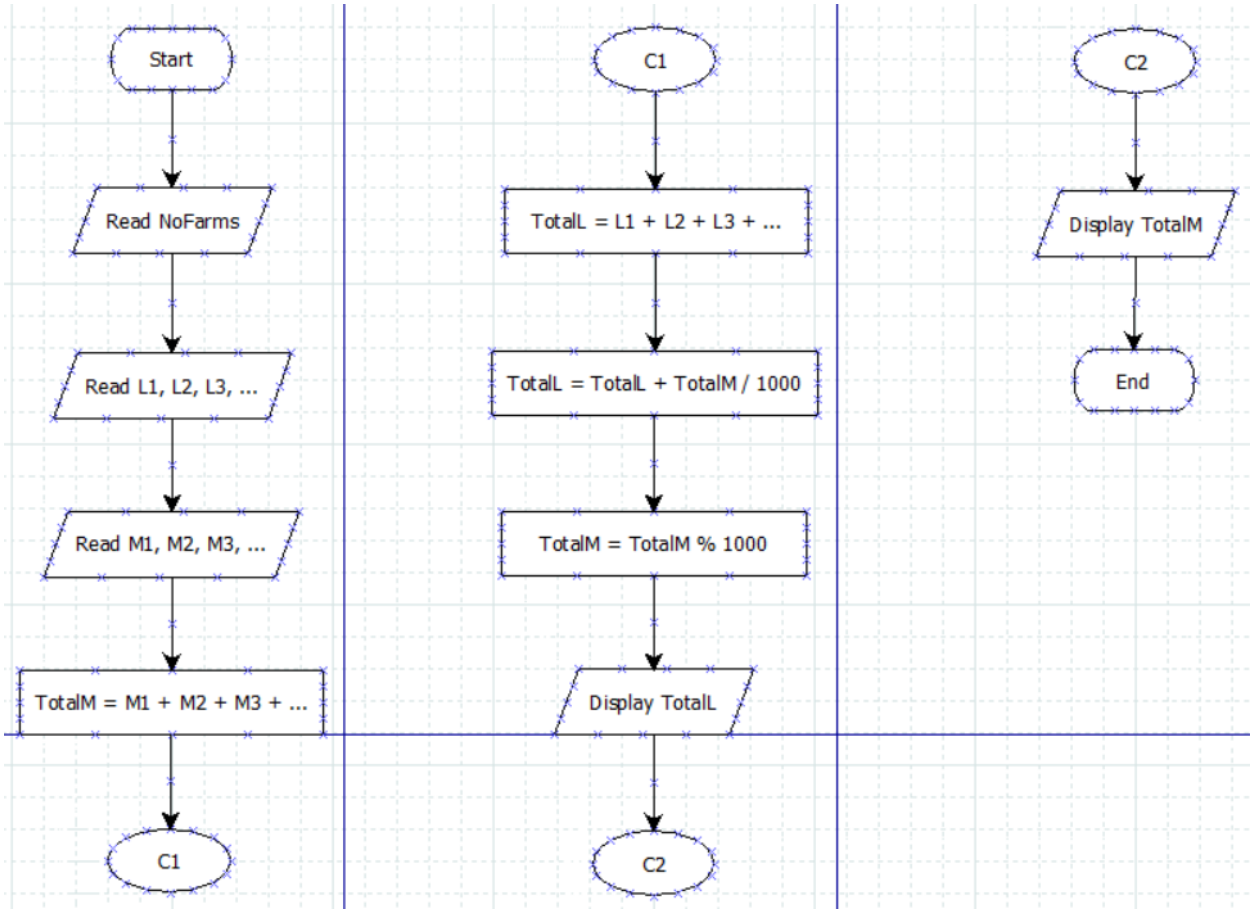
```
days = int(input("Enter days the fly has been alive: "))
seconds = days * 24 * 3600
print(f"The fly has lived for {seconds} seconds")
```

M12_CSQ2: Milk is collected for sales from nearest ‘n’ farms to the milk booth. Given the amount of milk from ‘n’ farms in litres and ml. Write a PAC chart, algorithm, and flowchart to compute total quantity of milk in the booth.

PAC:

Data	Processing	Output	Alternative Solutions
NoFarms = user input L1, L2, L3, … = user input M1, M2, M3, … = user input	TotalM = M1 + M2 + … TotalL = L1 + L2 + L3 + … TotalL = TotalL + TotalM / 1000 TotalM = TotalM % 1000	TotalL TotalM	Define NoFarms Define NoLiters, NoMillis

Flowchart:



Algorithm:

Read NoFarms

Read L1, L2, L3, …

Read M1, M2, M3, …

Calculate TotalM = M1 + M2 + …

Calculate TotalL = L1 + L2 + L3 + …

Calculate TotalL = TotalL + TotalM / 1000

Calculate TotalM = TotalM % 1000

Display TotalL

Display TotalM

Code:

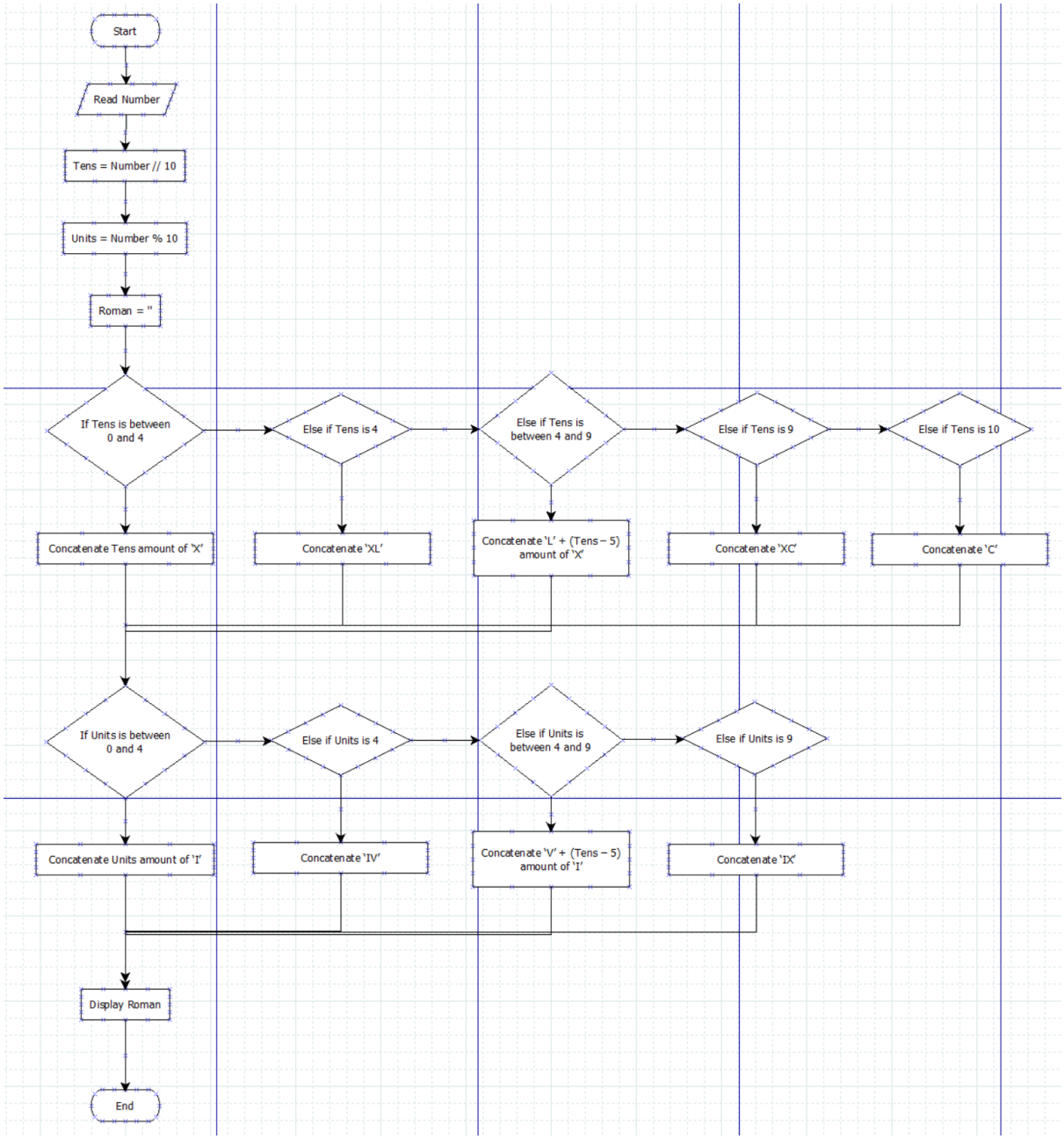
```
farms = int(input("Enter number of farms: "))
litre_sum = 0
milli_sum = 0
for i in range(farms):
    milk = input(f"Farm {i + 1}: ").split(" ")
    litre_sum += int(milk[0])
    milli_sum += int(milk[1])

# carrying over the extra milliliters to liters
litre_sum += int(milli_sum / 1000)
milli_sum %= 1000
print(f"Total is {litre_sum} liters and {milli_sum} milliliters")
```

M12_CSQ3: Write a PAC chart, algorithm, and flowchart for converting the given two-digit number into its corresponding Roman numeral.

PAC:

Data	Processing	Output	Alternative Solutions
Number = user input	Tens = Number // 10 Units = Number % 10 Roman = ‘’ If Tens is between 0 and 4, then concatenate Tens amount of ‘X’ Else if Tens is 4, concatenate ‘XL’ Else if Tens is between 4 and 9, then concatenate ‘L’ + (Tens – 5) amount of ‘X’ Else if Tens is 9, then concatenate ‘XC’ Else if Tens is 10, then concatenate ‘C’ If Units is between 0 and 4, then concatenate Units amount of ‘I’ Else if Units is 4, concatenate ‘IV’ Else if Units is between 4 and 9, then concatenate ‘V’ + (Tens – 5) amount of ‘I’ Else if Units is 9, then concatenate ‘IX’	Roman	Define Number

Flowchart:



Algorithm:

Read Number

Tens = Number // 10

Units = Number % 10

If Tens is between 0 and 4, then concatenate Tens amount of ‘X’

Else if Tens is 4, concatenate ‘XL’

Else if Tens is between 4 and 9, then concatenate ‘L’ + (Tens – 5) amount of ‘X’

Else if Tens is 9, then concatenate ‘XC’

Else if Tens is 10, then concatenate ‘C’

If Units is between 0 and 4, then concatenate Units amount of ‘I’

Else if Units is 4, concatenate ‘IV’

Else if Units is between 4 and 9, then concatenate ‘V’ + (Tens – 5) amount of ‘I’

Else if Units is 9, then concatenate ‘IX’

Display Roman

Code:

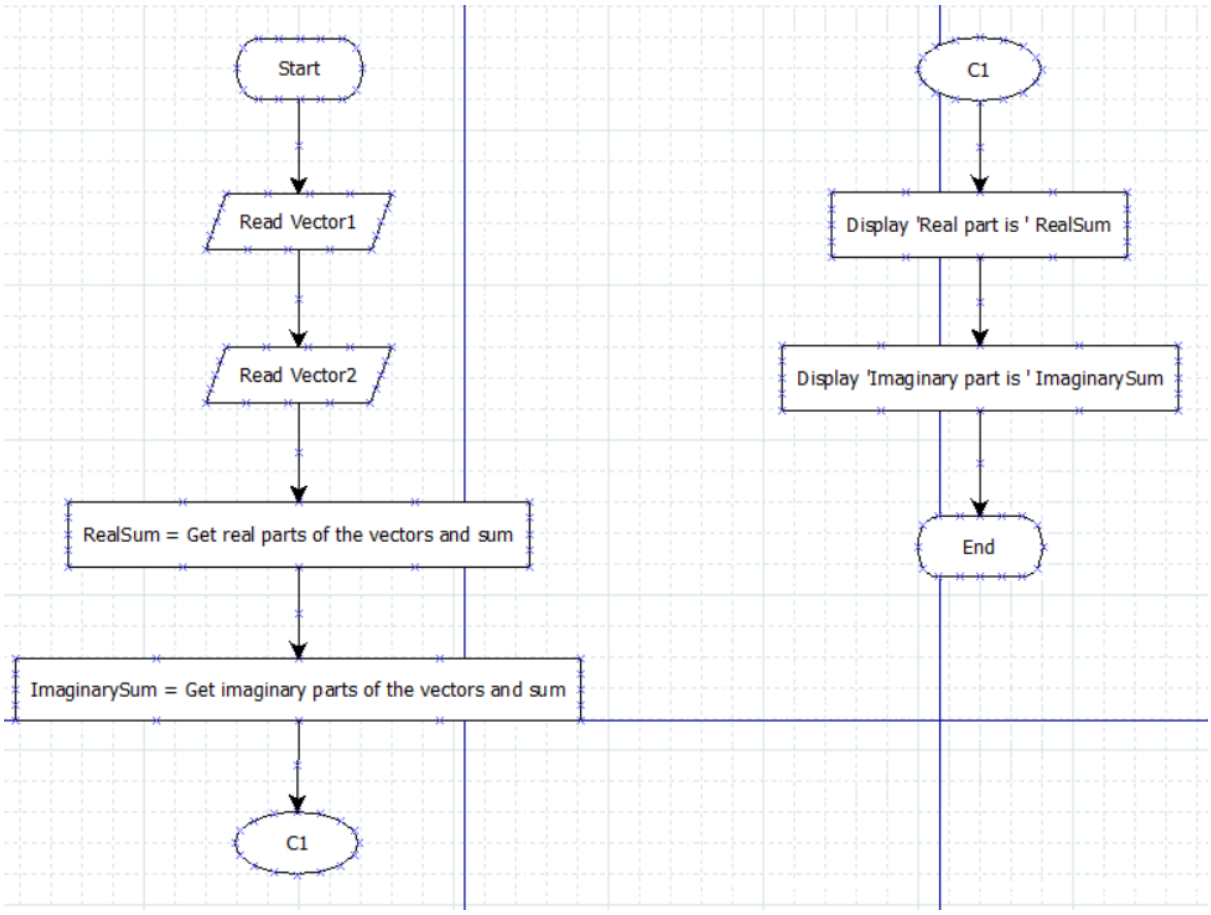
```
number = int(input("Enter number: "))
roman = ""

tens = number // 10
units = number % 10

if tens > 0 and tens < 4:
    roman += (tens * 'X')
elif number / 10 == 4:
    roman += 'XL'
elif tens > 4 and tens < 9:
    roman += 'L' + ((tens - 5) * 'X')
elif tens == 9:
    roman += 'XC'
elif tens == 10:
    roman += 'C'

if units > 0 and units < 4:
    roman += (units * 'I')
elif units == 4:
    roman += 'IV'
elif units > 4 and units < 9:
    roman += 'V' + ((units - 5) * 'I')
elif units == 9:
    roman += 'IX'

print(f"{number} in Roman numerals is {roman}")
```

M12_CSQ4: Write a PAC chart, Algorithm and Flowchart for adding two complex numbers, input two complex numbers and add the same to produce the result. After producing the result, print the real part and imaginary part separately.

PAC:

Data	Processing	Output	Alternative Solutions
Vector1 = user input Vector2 = user input	RealSum = Real parts of the vectors and sum ImaginarySum = Imaginary parts of the vectors and sum	'Real part is ' RealSum 'Imaginary part is ' ImaginarySum	Define Vector1 and Vector2

Flowchart:



Algorithm:

Read Vector1, Vector2

Calculate RealSum = Get real parts of the vectors and sum

Calculate ImaginarySum = Get imaginary parts of the vectors and sum

Display 'Real part is ' RealSum

Display 'Imaginary part is ' ImaginarySum

Code:

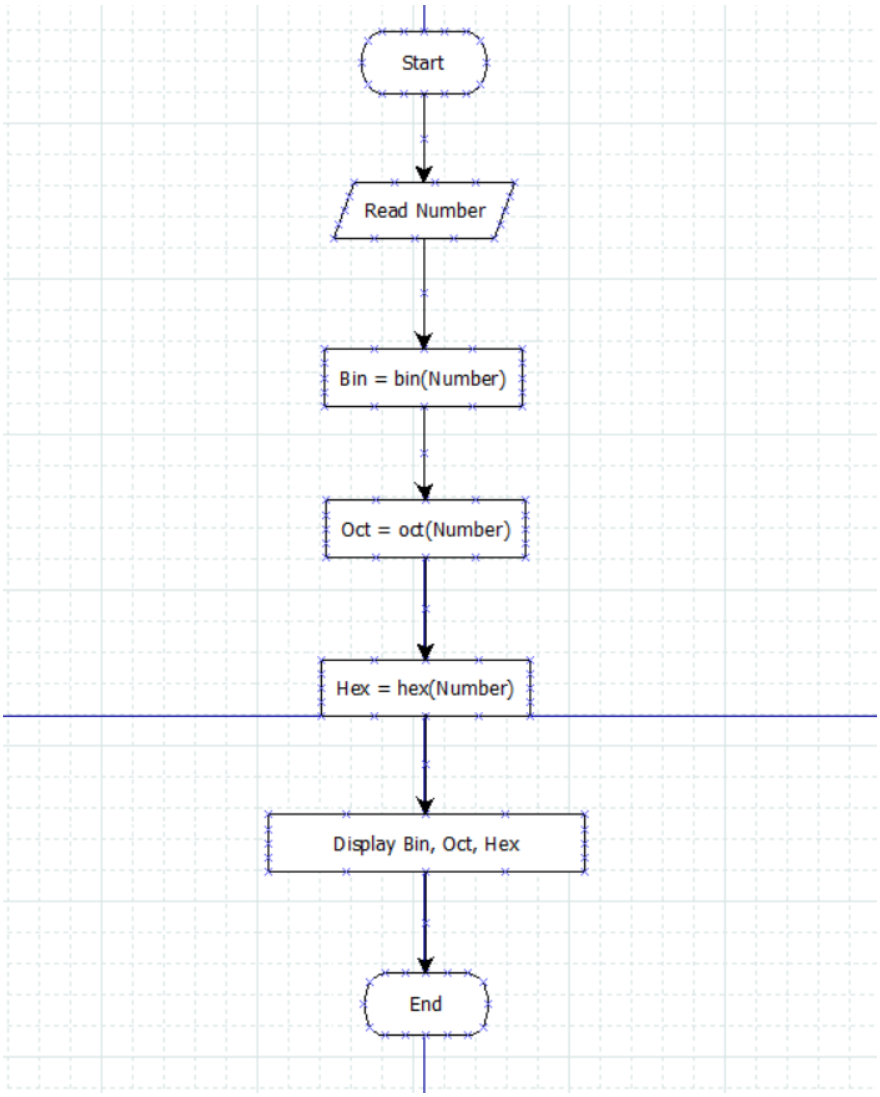
```
vector1 = complex(input("Enter vector 1: "))
vector2 = complex(input("Enter vector 2: "))

real_sum = vector1.real + vector2.real
imaginary_sum = vector1.imag + vector2.imag

print(f"Real part is: {int(real_sum)}")
print(f"Imaginary part is: {int(imaginary_sum)}")
```

M12_CSQ5: Write a PAC chart, Algorithm and Flowchart to convert the given integer to the corresponding binary, octal and hexadecimal values and print the same.

PAC:

Data	Processing	Output	Alternative Solutions
Number = user input	Python in built functions: Bin = bin(Number) Oct = oct(Number) Hex = hex(number)	Bin Oct Hex	Define Number Compute Bin, Oct, Hex manually

Flowchart:



Algorithm:

Read Number

Calculate Bin = bin(Number)

Calculate Oct = oct(Number)

Calculate Hex = hex(number)

Display Bin, Oct, Hex

Code:

```
num = int(input())

print(bin(num), oct(num), hex(num))
```