In [ ]:
```
Program 1(a):
a) Develop a program to read the student details like Name, USN, and Marks in three subjects. Display the studen
percentage with suitable messages.
```

In [2]:
```python
def input_student_details():
    name = input("Enter student's name: ")
    usn = input("Enter student's USN: ")
    marks = []

    for i in range(1, 4):
        mark = float(input(f"Enter marks for subject {i}: "))
        marks.append(mark)

    return name, usn, marks

def calculate_results(marks):
    total_marks = sum(marks)
    percentage = (total_marks / 300) * 100  # Assuming each subject is out of 100 marks
    return total_marks, percentage

def display_student_details(name, usn, total_marks, percentage):
    print("\n--- Student Details ---")
    print(f"Name: {name}")
    print(f"USN: {usn}")
    print(f"Total Marks: {total_marks}")
    print(f"Percentage: {percentage:.2f}%")

name, usn, marks = input_student_details()

total_marks, percentage = calculate_results(marks)

display_student_details(name, usn, total_marks, percentage)
```

```
--- Student Details ---
Name: Deen
USN: 1dt26cf017
Total Marks: 285.0
Percentage: 95.00%
```

In [ ]:
```
Program 1(b):
Develop a program to read the name and year of birth of a person.Display whether the person is a senior citizen
```

In [4]:
```python
from datetime import datetime

def input_person_details():
    name = input("Enter the person's name: ")
    year_of_birth = int(input("Enter the year of birth: "))
    return name, year_of_birth

def is_senior_citizen(year_of_birth):
    current_year = datetime.now().year
    age = current_year - year_of_birth
    return age >= 60

def display_result(name, is_senior):
    print("\n--- Result ---")
    if is_senior:
        print(f"{name} is a senior citizen.")
    else:
        print(f"{name} is not a senior citizen.")

name, year_of_birth = input_person_details()

is_senior = is_senior_citizen(year_of_birth)

display_result(name, is_senior)
```

```
--- Result ---
Sujaya is a senior citizen.
```

In [ ]:
```
Program 2(a):
Develop a program to generate Fibonacci sequence of length (N). Read N from the console.
```

In [5]:
```python
def generate_fibonacci(n):
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]

    fibonacci_sequence = [0, 1]
    for i in range(2, n):
        next_number = fibonacci_sequence[-1] + fibonacci_sequence[-2]
        fibonacci_sequence.append(next_number)

    return fibonacci_sequence

n = int(input("Enter the length of the Fibonacci sequence (N): "))

fibonacci_sequence = generate_fibonacci(n)

print(f"\nFibonacci sequence of length {n}:")
print(fibonacci_sequence)
```

```
Fibonacci sequence of length 10:
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

```
In [ ]: Program 2(b):
        Write a function to calculate factorial of a number. Develop a program to compute binomial coefficient (Given N
```

```
In [6]: def factorial (n):
            if n==0 or n==1:
                return 1
            else:
                return n * factorial(n-1)

        def binomial_coeffecient(n,r):
            return factorial(n)// (factorial(r) * factorial (n-r))

        n=int(input("Enter the value of N:"))
        r=int(input("Enter the valur of R:"))

        if r>n:
            print("Invalid input.R cannot be greater than N")
        else:
            result=binomial_coeffecient(n,r)
            print(f"\n Binomial Coeffecient C({n},{r})={result}")
```

```
 Binomial Coeffecient C(7,4)=35
```

```
In [ ]: Program 3:
        Read N numbers from the console and create a list. Develop a program to print mean, variance and standard devia
```

```
In [7]: import math

        def calculate_mean(numbers):
          return sum(numbers) / len(numbers)

        def calculate_variance(numbers, mean):
          return sum((x - mean) ** 2 for x in numbers) / len(numbers)

        def calculate_standard_deviation(variance):
          return math.sqrt(variance)

        n = int(input("Enter the number of elements (N): "))

        numbers = []
        for i in range(n):
          number = float(input(f"Enter number {i + 1}: "))
          numbers.append(number)

        mean = calculate_mean(numbers)
        variance = calculate_variance(numbers, mean)
        std_deviation = calculate_standard_deviation(variance)

        print(f"\nMean: {mean:.2f}")
        print(f"Variance: {variance:.2f}")
        print(f"Standard Deviation: {std_deviation:.2f}")
```

```
Mean: 20.00
Variance: 50.00
Standard Deviation: 7.07
```

```
In [ ]: Program 4:
        Read a multi-digit number (as chars) from the console. Develop a program to print the frequency of each digit w
```

```
In [8]: def calculate_digit_frequency(number):
          frequency = {}
          for digit in number:
            if digit.isdigit():
              frequency[digit] = frequency.get(digit, 0) + 1
          return frequency

        def display_digit_frequency(frequency):
```

```
  print("\nFrequency of each digit:")
  for digit, count in sorted(frequency.items()):
    print(f"Digit {digit}: {count} time(s)")

number = input("Enter a multi-digit number: ")

frequency = calculate_digit_frequency(number)

display_digit_frequency(frequency)
```

```
Frequency of each digit:
Digit 0: 1 time(s)
Digit 1: 3 time(s)
Digit 2: 3 time(s)
Digit 3: 1 time(s)
Digit 4: 2 time(s)
Digit 5: 1 time(s)
Digit 7: 2 time(s)
Digit 8: 1 time(s)
Digit 9: 3 time(s)
```

In [ ]:
```
Program 5:
Develop a program to print 10 most frequently appearing words in a text file.[Hint: Use dictionary with distinc
Sort the dictionary in the reverse order of frequency and display dictionary slice of first 10 items]
```

In [10]:
```
from collections import defaultdict
import string

def count_word_frequencies(file_path):
    word_frequency = defaultdict(int)
    with open(file_path, 'r') as file:
        for line in file:
            line = line.translate(str.maketrans('', '', string.punctuation)).lower()
            words = line.split()
            for word in words:
                word_frequency[word] += 1
    return word_frequency

def get_top_n_words(word_frequency, n=10):
    sorted_words = sorted(word_frequency.items(), key=lambda item: item[1], reverse=True)
    return sorted_words[:n]

file_path = input("Enter the path to the text file: ")

word_frequency = count_word_frequencies(file_path)

top_words = get_top_n_words(word_frequency)

print("\nThe 10 most frequently appearing words are:")
for word, freq in top_words:
    print(f"{word}: {freq} time(s)")
```

```
The 10 most frequently appearing words are:
non: 12 time(s)
quod: 11 time(s)
esse: 9 time(s)
est: 8 time(s)
cum: 8 time(s)
a: 7 time(s)
et: 6 time(s)
ut: 6 time(s)
ita: 6 time(s)
quae: 5 time(s)
```

In [ ]:
```
Program 6:
Develop a program to sort the contents of a text file and write the sorted contents into a separate text file.
sort(), append(), and file methods open(), readlines(), and write()].
```

In [11]:
```
def sort_file_contents(input_file_path, output_file_path):
    try:
        with open(input_file_path, 'r') as file:
            lines = file.readlines()

        stripped_lines = [line.strip() for line in lines]
        stripped_lines.sort()

        with open(output_file_path, 'w') as file:
            for line in stripped_lines:
                file.write(line + '\n')
        print(f"Sorted contents have been written to '{output_file_path}'")
    except FileNotFoundError:
        print(f"The file '{input_file_path}' was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")
```

```
input_file_path = input("Enter the path to the input text file: ")

output_file_path = input("Enter the path to the output text file: ")

sort_file_contents(input_file_path, output_file_path)
```

```
Sorted contents have been written to 'C:\Users\hp\Downloads\output file.txt'
```

In [ ]:
```
Program 7:
Develop a program to backing Up a given Folder (Folder in a current working directory) into a ZIP File by using
```

In [16]:
```
import os
import zipfile
from datetime import datetime

def backup_folder_to_zip(folder_name):
    folder_path = os.path.abspath(folder_name)
    timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
    zip_filename = f"{folder_name}_{timestamp}.zip"
    with zipfile.ZipFile(zip_filename, 'w', zipfile.ZIP_DEFLATED) as backup_zip:
        for foldername, subfolders, filenames in os.walk(folder_path):
            backup_zip.write(foldername, os.path.relpath(foldername, folder_path))
            for filename in filenames:
                file_path = os.path.join(foldername, filename)
                backup_zip.write(file_path, os.path.relpath(file_path, folder_path))
    print(f"Backup complete! The folder '{folder_name}' has been backed up into '{zip_filename}'.")

folder_name = input("Enter the name of the folder to back up: ")

if os.path.isdir(folder_name):
    backup_folder_to_zip(folder_name)
else:
    print(f"The folder '{folder_name}' does not exist in the current working directory.")
```

```
The folder 'DSA' does not exist in the current working directory.
```