✅ Procedure to Repeatedly Prompt the User Until a Valid Integer is Entered:

1. **Start a loop** that continues until the user gives valid input.
2. **Prompt the user** to enter a number.
3. **Attempt to convert** the user's input to an integer.
4. **Check if the conversion is successful:**
    - If it is successful, **accept the input** and **exit the loop**.
    - If it is not successful (i.e., the input is invalid), **display an error message**.
5. **Repeat the process** until a valid integer is entered.

✅ Procedure to Find the Most Frequently Occurring Number:

1. **Start with a list** of numbers as input.
2. **Create a way to count** how many times each number appears in the list (this is called frequency counting).
3. **Go through each number** in the list one by one.
4. **For each number:**
    - Increase its count if it has already appeared.
    - If it's the first time, start its count from 1.
5. **After processing all numbers**, check which number has the **highest count**.
6. **Return or display** the number with the highest frequency as the most frequent number.
7. (Optional) If there are multiple numbers with the same highest frequency, decide how to handle it — for example, return all of them or just the first one.

✅ Procedure to Check if Two Strings Are Anagrams:

1. **Take two input strings** that need to be compared.

2. **Remove any spaces** and **convert both strings to the same case** (either lowercase or uppercase) to ensure consistency.
3. **Check the length** of both strings:
   - If the lengths are not equal, they **cannot** be anagrams.
4. **Count the frequency** of each character in both strings.
5. **Compare the character counts**:
   - If both strings have the **same characters with the same frequency**, they are **anagrams**.
   - Otherwise, they are **not anagrams**.

✅ Procedure to Count Vowels in a String:

1. **Take the input string** (sentence, word, or paragraph).
2. **Convert the string to the same case** (lowercase or uppercase) to simplify comparison.
3. **Define the set of vowels**: `a, e, i, o, u`.
4. **Go through each character** in the string one by one.
5. **Check if the character is a vowel**:
   - If yes, **increase the vowel count by 1**.
6. **Continue this process** until all characters are checked.
7. **Return or display the final count** of vowels.

✅ Procedure to Reverse the Order of Words in a Sentence:

1. **Take the input sentence** as a string.
2. **Split the sentence into words** using spaces as the separator.
3. **Store the words in a list or sequence** for processing.
4. **Reverse the order** of the words in the list.
5. **Join the reversed words** back into a sentence using spaces.

6. **Return or display** the new sentence with words in reversed order.

✅ Example:

- Input: `"Data Science is amazing"`
- Output: `"amazing is Science Data"`

✅ Procedure to Find the Missing Number in a Sequence from 1 to n:

1. **Take the input list** that contains `n - 1` numbers.
2. **Determine the expected total count** of numbers (`n`).
   - You can get it by adding 1 to the length of the list: `n = length of list + 1`.
3. **Calculate the expected sum** of numbers from 1 to n using the formula:
   - `Expected Sum = n × (n + 1) ÷ 2`.
4. **Calculate the actual sum** of the numbers present in the list.
5. **Subtract the actual sum from the expected sum**:
   - `Missing Number = Expected Sum - Actual Sum`.
6. **Return or display** the missing number.

✅ Example:

- If list = `[1, 2, 4, 5]`, then `n = 5`
- Expected Sum = `1 + 2 + 3 + 4 + 5 = 15`
- Actual Sum = `1 + 2 + 4 + 5 = 12`
- Missing Number = `15 - 12 = 3`

✅ Procedure to Process ATM Withdrawal Based on Balance:

1. **Take the current account balance** as input.
2. **Take the withdrawal amount** as input.
3. **Compare the withdrawal amount with the account balance**:
   - If the **withdrawal amount is less than or equal to the balance**, allow the withdrawal.
   - If the **withdrawal amount is greater than the balance**, deny the withdrawal.
4. **If withdrawal is allowed**:
   - Subtract the withdrawal amount from the account balance.
   - Display a success message and show the updated balance.
5. **If withdrawal is denied**:
   - Show an error message indicating **insufficient balance**.

✅ Procedure to Check for Duplicate Values in a List:

1. **Take the input list** containing the data entries.
2. **Create an empty set** to store unique values.
3. **Go through each item** in the list one by one.
4. **For each item:**
   - Check if the item is already in the set:
     - If **yes**, a **duplicate is found** — stop and return that the list contains duplicates.
     - If **no**, add the item to the set and continue.
5. **If the entire list is processed** and no duplicates are found, confirm that the list has **no duplicate values**.

✅ Procedure to Sum Digits of an Integer:

1. **Take the input number** (integer).
2. **Convert the number to a string** (optional) to process each digit easily, or work with the number mathematically.
3. **Initialize a variable to store the sum** of digits, starting at 0.
4. **Go through each digit** of the number:
   - If using string method, process each character.
   - If using math, extract digits one by one (e.g., using modulus and division).
5. **Convert each digit back to an integer** (if using string method).
6. **Add the digit to the sum variable**.
7. **Continue until all digits are processed**.
8. **Return or display the final sum** of the digits.

✅ Procedure to Check if a Sentence is a Pangram:

1. **Take the input sentence** as a string.
2. **Convert the sentence to lowercase** to ignore case differences.
3. **Create a set of all alphabets** (a to z).
4. **Create a set of all letters found** in the sentence.
5. **Compare the two sets**:
   - If the set of letters found contains **all alphabets**, the sentence is a pangram.
   - Otherwise, it is not a pangram.