Revision Notes: Problem Solving Review 1

Validation and Verification

- **Validation** checks ensure that data input is reasonable and sensible. For example, if a program asks for a user's age, a validation check might ensure that the input is a positive number. Here are some specific types of validation checks:
- **Examples of Validation in Pseudocode**
  - **Range Check**

    Description: Checks if a value falls within a specified range.

    Example Scenario: Ensuring an exam mark entered is between 0 and 100.

    Pseudocode:

    ```
    REPEAT
      INPUT Mark
      IF Mark < 0 OR Mark > 100 THEN
        OUTPUT "Invalid mark. Please enter a value between 0 and 100."
      ENDIF
    UNTIL Mark >= 0 AND Mark <= 100
    ```

  - **Presence Check**

    Description: Checks if some data has been entered.

    Example Scenario: Making sure a user enters their name in a registration form.

    Pseudocode:

    ```
    REPEAT
      INPUT Name
      IF Name = "" THEN
        OUTPUT "Name field cannot be empty. Please enter your name."
      ENDIF
    UNTIL Name <> ""
    ```

- **Length Check**

  Description: Checks for a maximum number of characters in the data entered.

  Example Scenario: Limiting a password to a maximum of 12 characters.

  Pseudocode:

  ```
  REPEAT
    INPUT Password
    IF LENGTH(Password) > 12 THEN
      OUTPUT "Password too long. Must be 12 characters or less."
    ENDIF
  UNTIL LENGTH(Password) <= 12
  ```

- **Type Check**

  Description: Checks that the characters entered are all numbers.

  Example Scenario: Verifying that an input for age only contains numerical digits.

  Pseudocode:

  ```
  FUNCTION IsNumber(input) : BOOLEAN
    //This function would use string manipulation techniques to
    //check if all characters in the 'input' string are digits 0-9.
    //Implementation details are omitted for brevity.
  ENDFUNCTION


  REPEAT
    INPUT Age
    IF NOT IsNumber(Age) THEN
      OUTPUT "Invalid age. Please enter numbers only."
    ENDIF
  UNTIL IsNumber(Age)
  ```

- **Combined Validation in a Loop**

  The sources offer an example of a validation check within a loop to ensure a value is within a given range.

  ```
  PRINT "Input a value between 0 and 100 inclusive"
  INPUT Value
  WHILE Value < 0 OR Value > 100
     PRINT "Invalid value, try again"
     INPUT Value
  ENDWHILE
  PRINT "Accepted: ", Value
  ```

  This code snippet combines a range check with a loop to repeatedly prompt the user for input until a valid value is entered.

- **Verification** checks ensure that data is entered correctly. For example, a verification check might require a user to enter their password twice to ensure that they have typed it correctly.

Programming Constructs

Sequence

- Sequence is a fundamental programming construct. It refers to the execution of instructions one after the other in the order they appear in the program.

Selection

- Selection statements, allow the program to choose different paths of execution based on certain conditions.
- Examples of selection statements include "CASE" statements and "IF" statements.

Repetition (Loops)

- Repetition statements, also known as loops, allow a block of code to be executed repeatedly until a certain condition is met.
- There are three loop structures:
  - FOR...TO...NEXT loop: This loop structure iterates a predetermined number of times.

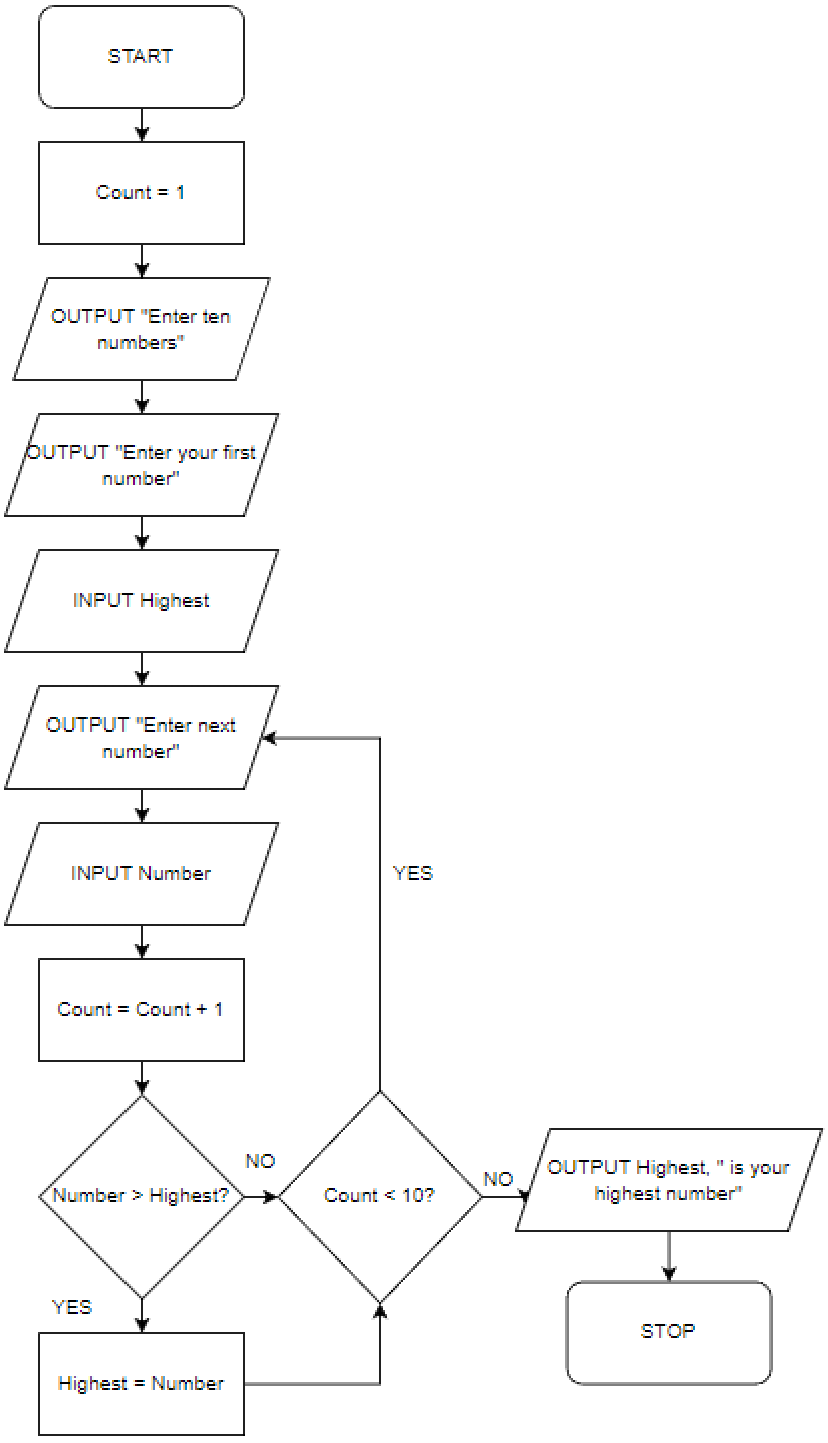
    - WHILE...ENDWHILE loop: This loop continues to iterate as long as a specified condition remains true.
    - REPEAT...UNTIL loop: This loop structure continues to iterate until a given condition becomes true.

Data Handling

- **Constants:** Values that do not change during the execution of a program. They are used to store fixed values that are used multiple times within the code, improving readability and maintainability.
- **Variables:** Storage locations that hold values that can change during program execution. They allow the program to work with and manipulate data.
- **Arrays:** Used to store collections of data of the same data type under a single identifier. Each element in an array can be accessed using its index.

Section 5: Flowcharts and Trace Tables

- **Flowcharts** visually represent the flow of a program's execution using different shapes to denote different actions or decisions.
- **Trace tables** are used to test algorithms and sections of code by tracking the values of variables step-by-step as the code executes with different inputs.
    - Trace Table Walkthrough
        1. Below is a flowchart to determine the highest number of ten user entered numbers
        2. The algorithm prompts the user to enter the first number which automatically becomes the highest number entered
        3. The user is then prompted to enter nine more numbers. If a new number is higher than an older number then it is replaced
        4. Once all ten numbers are entered, the algorithm outputs which number was the highest
        5. Example test data to be used is: 4, 3, 7, 1, 8, 3, 6, 9, 12, 10


START
Count = 1
OUTPUT "Enter ten numbers"
OUTPUT "Enter your first number"
INPUT Highest
OUTPUT "Enter next number"
INPUT Number
Count = Count + 1
Number > Highest?
NO
YES
Highest = Number
Count < 10?
YES
NO
OUTPUT Highest, " is your highest number"
STOP

Trace table for Figure 1: Highest number			
Count	Highest	Number	Output
1			Enter ten numbers
	4		Enter your first number
2		3	Enter your next number
3	7	7	
4		1	
5	8	8	
6		3	
7		6	
8	9	9	
9	12	12	
10		10	12 is your highest number