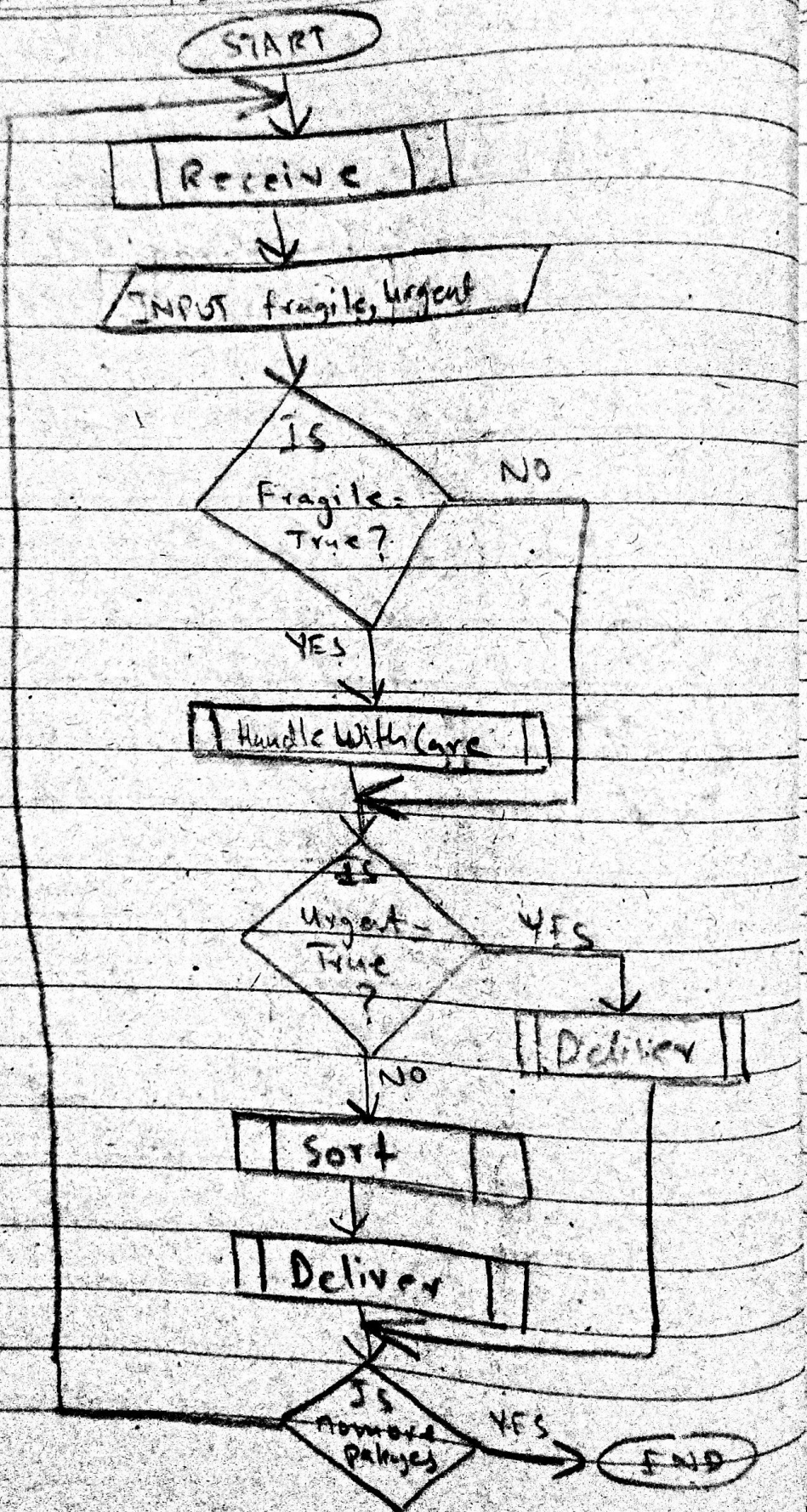
PF - LAB 02

PJ = 4/18

Q2) Flowchart

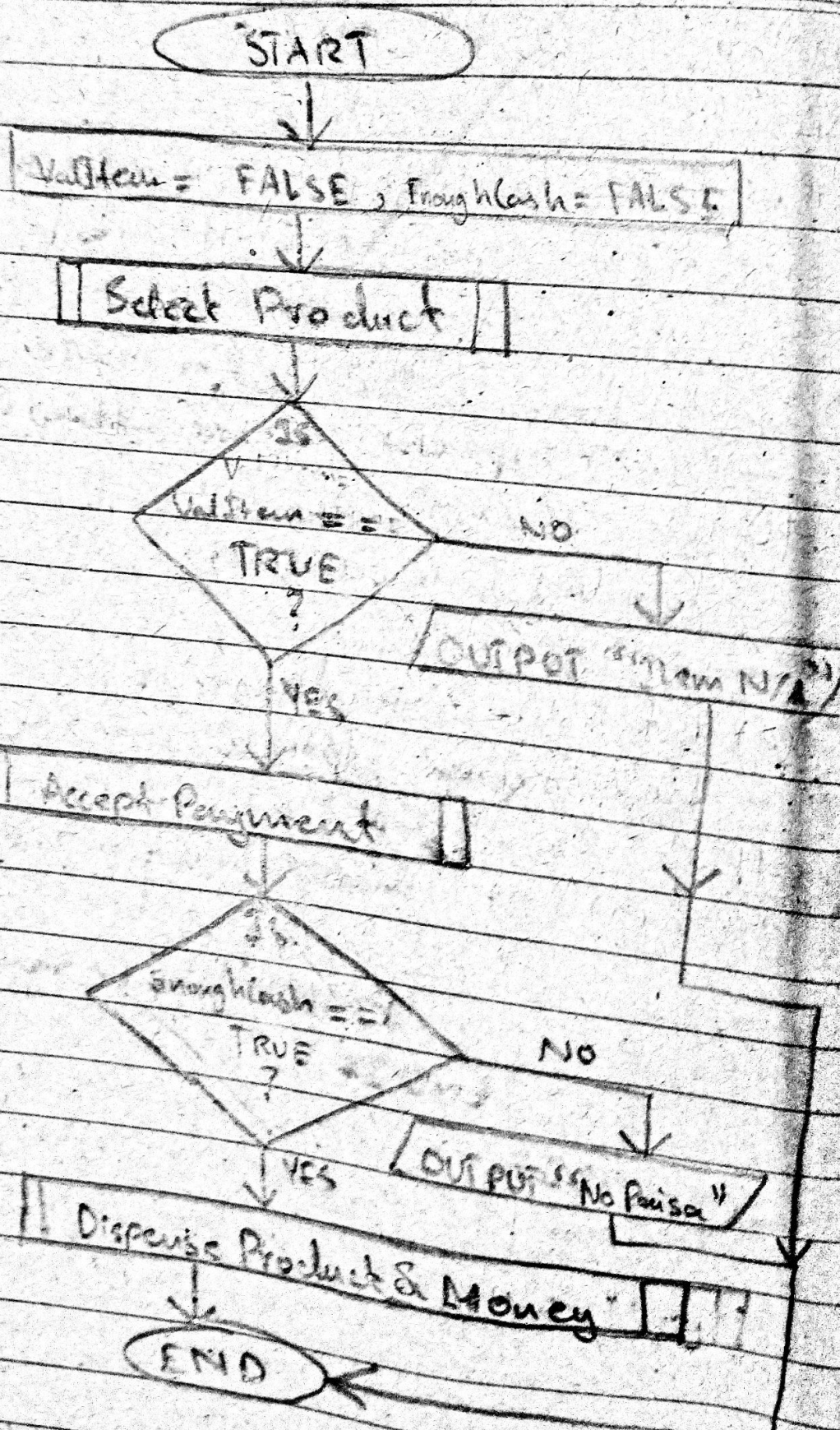


Q1) PSEUDOCODE

```
START
NOMORE PACKAGES = FALSE

REPEAT
    CALL (Receive)
    INPUT Fragile, Urgent

    IF Fragile == TRUE THEN
        CALL (Handle With Care)
    ENDIF
    IF Urgent == TRUE THEN
        CALL( Deliver)
    ELSE
        CALL (Sort)
        CALL (Deliver)
    ENDIF
UNTIL NOMOREPACKAGES == TRUE
GO TO REPEAT

STOP
```

Flowchart

Q2)



(Q2) PSEUDOCODE

```
START

ValItem= FALSE          // Declaration and Initialization
EnoughCash= FALSE       // of variables

CALL (Select Product) //Calling Module
//Module updates the above Initialized variables
IF ValItem== FALSE THEN
    PRINT "Item N/A"
ELSE
    Call (Accept Payment)
// Accept Payment updates EnoughCash variable
    IF EnoughCash == TRUE THEN
        CALL (Dispense Product and Money)
    ELSE
        PRINT "No Paisa"
    END IF
ENDIF

STOP
```

Q2) Write Pseudocode to find the smallest number among three given variables. Implement a decesion-making structure to compare the variables.

```
START

DECLARE  Num1, Num2, Num3 : INTEGER
INPUT  Num1, Num2, Num3

IF  Num1 > Num2  THEN
    IF  Num3 > Num2  THEN
        Print "The Smallest No# is", Num2
    ELSE
        Print "The Smallest No# is", Num3
    ENDIF
ELSEIF  Num1 > Num3
    Print "The Smallest No# is", Num1
ELSE
    Print "The Smallest No# is" Num

STOP
```

Q2) Create Pseudocode to subtract two numbers without using the (−) operator. (Hint: use addition and complement techniques).

```
START

DECLARE Ans, N1, N2 : INTEGER
INPUT N1, N2

Ans = (NOT N2) + 1
Ans = N1 + Ans

PRINT "The Solution is", Ans

STOP
```

Q3) Desktop Pseudocode for a basic calculator that performs multiplication and division. The pseudocode should prompt the user for 2 nos and an operator, then display the result of the operation.

```
START

DECLARE  Num1, Num2, Ans : INTEGER
DECLARE  Op              : CHAR
PRINT "Number 1"
INPUT  Num1
PRINT "Number 2"
INPUT  Num2
PRINT "Enter Opcode x OR /"
INPUT  Op

IF  Op == '*'  THEN
        Ans = Num1 * Num2
        PRINT "Solution is", Ans
ELSEIF  Op == '/' THEN
        IF Num2 == 0 THEN
                PRINT "Invalid"
        ELSE
                Ans = Num1 / Num2
                PRINT "Solution is", Ans
        ENDIF
ELSE
        PRINT "Enter valid opcode"
ENDIF
STOP
```

(Q1) Write an algorithm to determine whether a number is a prime number. The algorithm should iterate through possible divisors other than and determine if the number has any divisors other than 1 and itself

Ans)

~~START~~

Step 1: Prompt text to ask user for an integer input that is positive also.

Step 2: READ number

Step 3: Check if the number is not Zero (0), then only proceed.

Step 4: Iterate from 1 to the (number) positive Integer (entered by the user) with a step of 1.

Step 5: Check if ~~the~~ ~~Positive~~ number modulo 2, returns a Zero (0), if ~~Step 6~~ it does GoTo Step 6, else GoTo Step 4.

Step 6: Compare ~~Check~~ if the value of the Counter variable used for iteration is not equals to the number and 1 is

not equals to the number, it this holds true, GO TO Step 7; else, GO TO Step 4.

Step 7: Increment the Counter variable used for iteration in the loop by 1

Step 7: Increment P_NUM COUNT (a seperate variable initialized to zero at the start) with 1.

Step 8. Once the loop ends, make a comparison for the P_NUMCOUNT value if it is greater equals to zero, then GO TO Step 9, otherwise GO TO Step 10.

Step 9: Display "Number is a Prime No#"

Step 10: Display "Number is not Prime"

Step 11: EXIT.

Q2)

Step 1: READ DayNum
Step2: Set DayInt to (DayNum Modulo 7)
Step3: If DayInt is equals to 1
then Set Day to Monday
Step4: If DayInt is equals to 2
then Set Day to Tuesday
Step5: If DayInt is equals to 3
then Set Day to Wednesday
Step6: If DayInt is equals to 4
then Set Day to Thursday
Step7: If DayInt is equals to 5
then Set Day to Friday.
Step8: If DayInt is equals to
6 then Set Day to Saturday.
Step9: If DayInt is equals to
0 then Set Day to Sunday.
Step10: Print "The day is", Day
Step11: Exit

Q3) Develop an algorithm for a program that takes two nos as input and finds the greatest Common Divisor (GCD) of the two numbers using the Euclidean algorithm.

Step1: Prompt the user to input any two numbers from the set of natural/ non-zero numbers, with Num being larger.

Step2: READ Num1, Num2

Step3: Check if any of the two numbers is equals to Zero; if yes GoTo Step #1, else GoTo Step 4.

Step4: Check if Num1 is smaller than Num2, if yes then GoTO step5, else GoTo Step8

Step5: Assign value of Num1 to Temp (a temporary variable).

Step6: Assign value of Num2 to Num1

Step7: Assign value of Temp to Num2.

Step8: Using a pre-conditional loop Check / look for when the value of Num2 equates to Zero, if true then iterating, GoTO Step 12. then keep on

Step9: Divide Num1 by Num2 to obtain and store the remainder in a variable (Rem), via the usage of modulo operator

Step10: Assign value of Num2 to Num1

Step11: Assign value of Rem to Num2, GO TO Step 8

Step12: Display / OUTPUT / PRINT the value stored in Num1, with an appropriate prompt, indicating this is the Greatest Common Divisor (GCD).

Step13: EXIT