Algorithm

Second Course-First Stage

Lecturer

Ferman Ali Ahmed

Department of Mathematics -College of Education

University of Salahaddin

Contents

1 INTRODUCTION TO PROGRAMMING

INTRODUCTION

A computer program is a sequential set of instructions written in a computer language that is used to direct the computer to perform a specific task of computation.

Observe that the definition demands that any set of instructions must be such that the tasks will usually be performed sequentially unless directed otherwise. Each instruction in the set will express a unit of work that a computer language can support. In general, high level languages, also known as 3GLs, support one human activity at a time. For example, if a computational task involves the determination of the average of three numbers, then it will require at least three human activities, *viz.*, getting the numbers, obtaining the sum of the numbers, and then obtaining the average. The process will therefore require three instructions in a computer language. However, it can be done using two instructions, also: first by obtaining the numbers and second by obtaining the sum and the average.

The objective of programming is to solve problems using computers quickly and accurately.

FLOWCHARTING AND ALGORITHMS

Flowcharts may be classified into two categories:

(i) Program Flowchart **(ii)** System Flowchart

Program flowcharts act like mirrors of computer programs in terms of flowcharting symbols. They contain the steps of solving a problem unit for a specific result.

System flowcharts contain the solutions of many problem units together that are closely related to each other and interact with each other to achieve a goal. We will first focus on program flowcharts.

A *program flowchart* is an extremely useful tool in program development. First, any error or omission can be more easily detected from a program flowchart than it can be from a program because a program flowchart is a pictorial representation of the logic of a program. Second, a program flowchart can be followed easily and quickly. Third, it serves as a type of documentation, which may be of great help if the need for program modification arises in future.

The following five rules should be followed while creating program flowcharts.

- Only the standard symbols should be used in program flowcharts.
- The program logic should depict the flow from top to bottom and from left to right.
- Each symbol used in a program flowchart should contain only one entry point and one exit point, with the exception of the decision symbol. This is known as the *single rule*.
- The operations shown within a symbol of a program flowchart should be expressed independently of any particular programming language.
- All decision branches should be well-labeled.

The following are the standard symbols used in program flowcharts:

Terminal: used to show the beginning and end of a set of computer-related processes

Input/Output: used to show any input/output operation

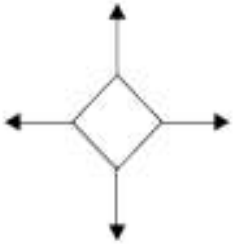
Computer processing: used to show any processing performed by a computer system

Predefined processing: used to indicate any process not specially defined in the flowchart

Comment: used to write any explanatory statement required to clarify something

Flow line: used to connect the symbols

Document Input/Output: used when input comes from a document and output goes to a document.



Decision: used to show any point in the process where a decision must be made to determine further action

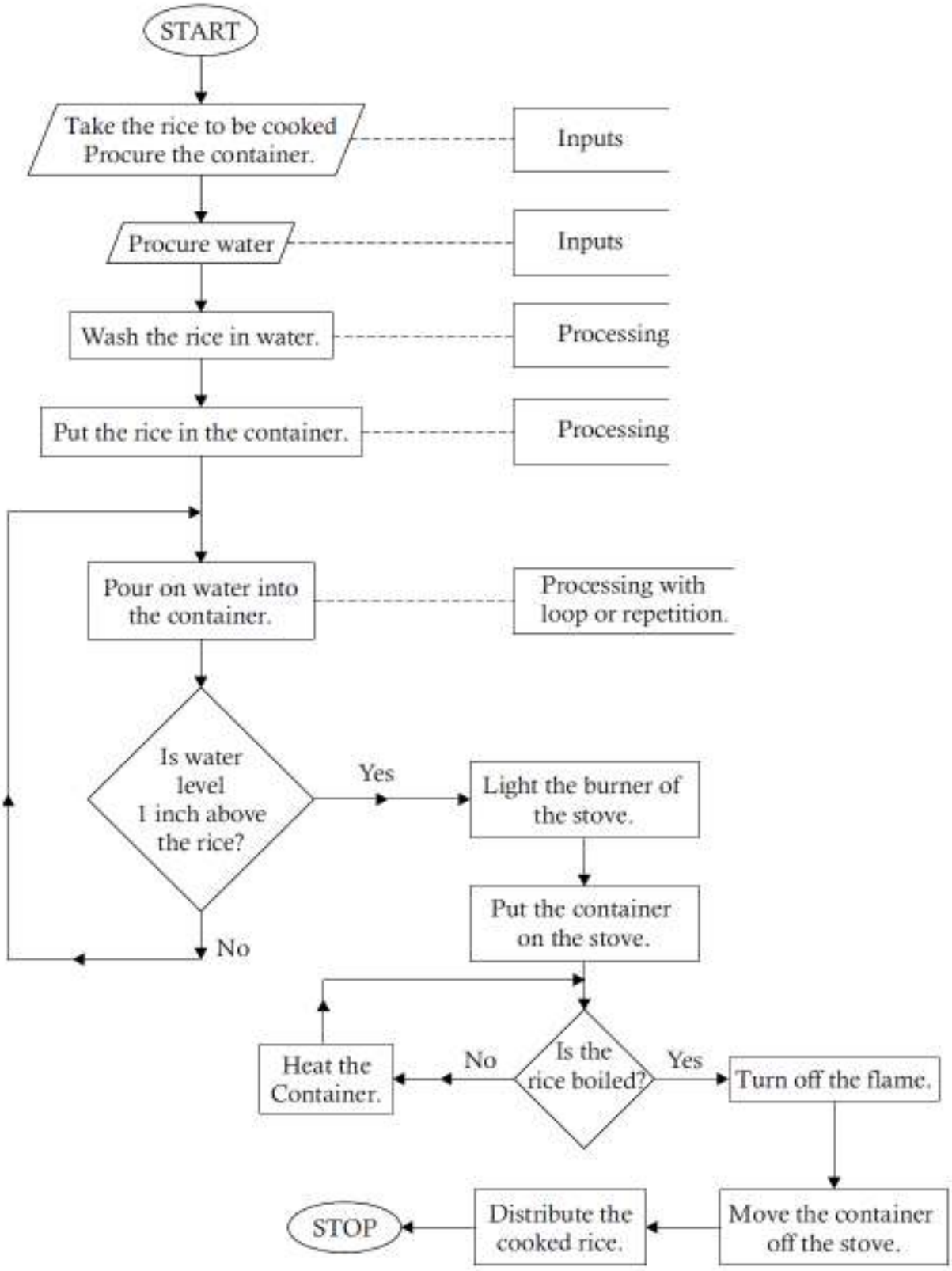
On-page connector: used to connect parts of a flowchart continued on the same page

Off-page connector: used to connect parts of a flowchart continued to separate pages

Flowcharts can be used to show the sequence of steps for doing any job. A set of simple operations involving accepting inputs, performing arithmetic operation on the inputs, and showing them to the users demonstrate the *sequence logic structure* of a program. The following flowchart shows the steps in cooking rice and then utilizing the cooked rice.

The algorithm for the flowchart about cooking rice is as follows:

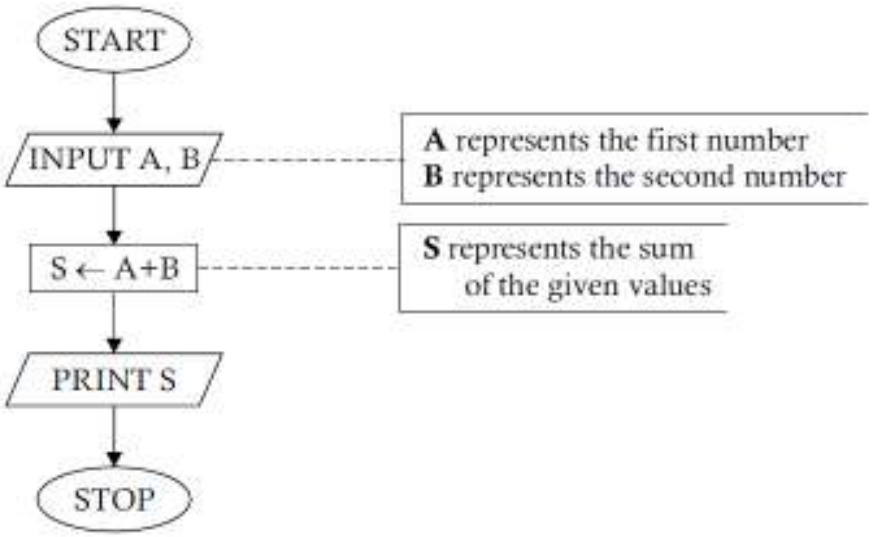
Step 1. Take the rice to be cooked.
Step 2. Procure the container.
Step 3. Procure the water.
Step 4. Wash the rice in the water.
Step 5. Put the rice into the container.
Step 6. Pour water into the container.
Step 7. IF WATER LEVEL = 1 INCH ABOVE THE RICE
THEN GOTO STEP 8
ELSE GOTO STEP 6
ENDIF


START
Take the rice to be cooked
Procure the container.
Inputs
Procure water
Inputs
Wash the rice in water.
Processing
Put the rice in the container.
Processing
Pour on water into the container.
Processing with loop or repetition.
Is water level 1 inch above the rice?
Yes
No
Light the burner of the stove.
Put the container on the stove.
Is the rice boiled?
No
Heat the Container.
Yes
Turn off the flame.
Move the container off the stove.
Distribute the cooked rice.
STOP

Step 8. Light the burner on the stove.
Step 9. IF THE RICE IS BOILED
THEN GOTO STEP 12
ELSE GOTO STEP 10
ENDIF
Step 10. Heat the container.
Step 11. Go to step 9.
Step 12. Turn off the flame.
Step 13. Move the container off the stove.
Step 14. Distribute the cooked rice.
Step 15. STOP.

The program logic structure illustrated in the flowcharts of this chapter is the sequence logic structure.

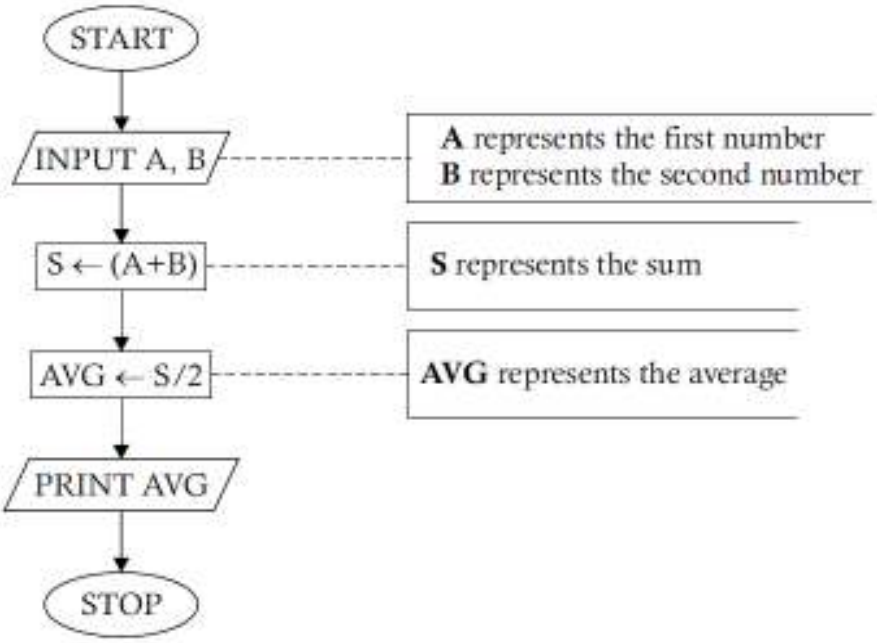
Problem 1.1. *Draw a flowchart to show how the sum of two numbers can be obtained.*



The following algorithm shows the desired procedure:

Step 1. INPUT TO A, B
Step 2. S ← A+B
(Store the sum of the values in A and B in S)
Step 3. PRINT S
(Show the sum obtained in Step 2)
Step 4. STOP

Problem 1.2. *Construct a flowchart to show the procedure for obtaining the average of two given numbers.*



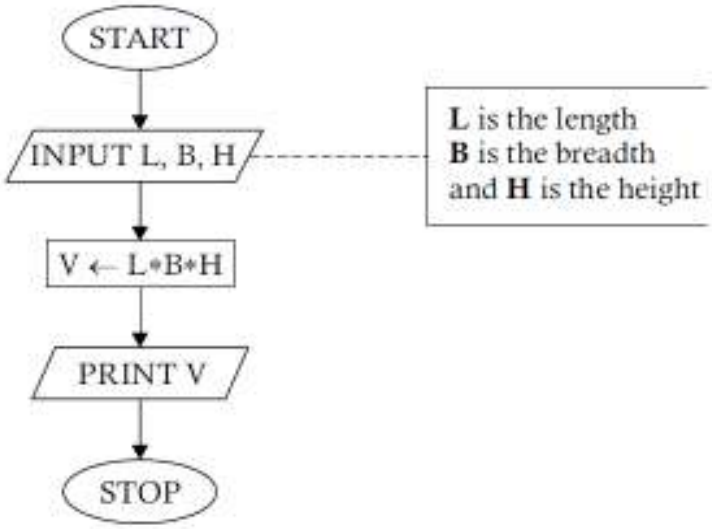
Task Analysis. From the concept of determining the average of two given numbers, we know that the given numbers must be added together to obtain the sum first; the sum is then divided by 2 to obtain the average. The flowchart for Problem 1.2 illustrates this idea.

The algorithm corresponding to Problem 1.2 is shown below:

Step 1. INPUT TO A, B
Step 2. S ← A + B
(Store the sum of the values in A and B and store in S)
Step 3. AVG ← S/2
(Compute the average)
Step 4. PRINT AVG (Show the average)
Step 5. STOP

Problem 1.3. *Construct a flowchart to show how to obtain the volume of a rectangular box.*

Task Analysis. We know that the formula to determine the volume of a rectangular box is Volume = Length × Breadth × Height. To determine the volume of a rectangular box, we need to know the length, breadth, and height of the box. When these values are multiplied together, the product represents the desired volume.

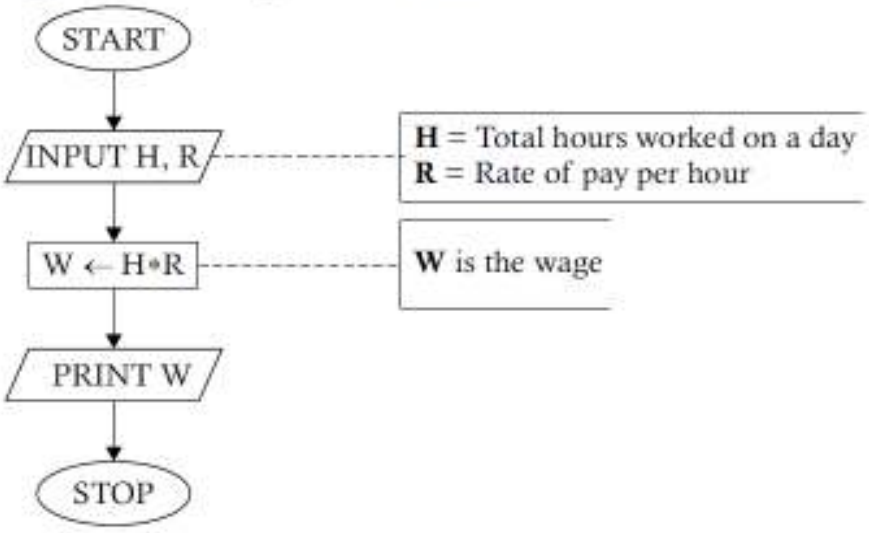


The algorithm for the solution of Problem 1.3 is given below:

Step 1. INPUT TO L, B, H
Step 2. COMPUTE V ← L*B*H
Step 3. PRINT V
Step 4. STOP

Problem 1.4. *Construct a flowchart to show how to obtain the daily wage of a worker on the basis of the hours worked during the day.*

Task Analysis. The daily wage depends on two factors: the hours worked and hourly rate of pay. When the hours worked is multiplied by the rate of pay, the product represents the wage of the worker.



The algorithm for the solution of Problem 1.4. is given below:

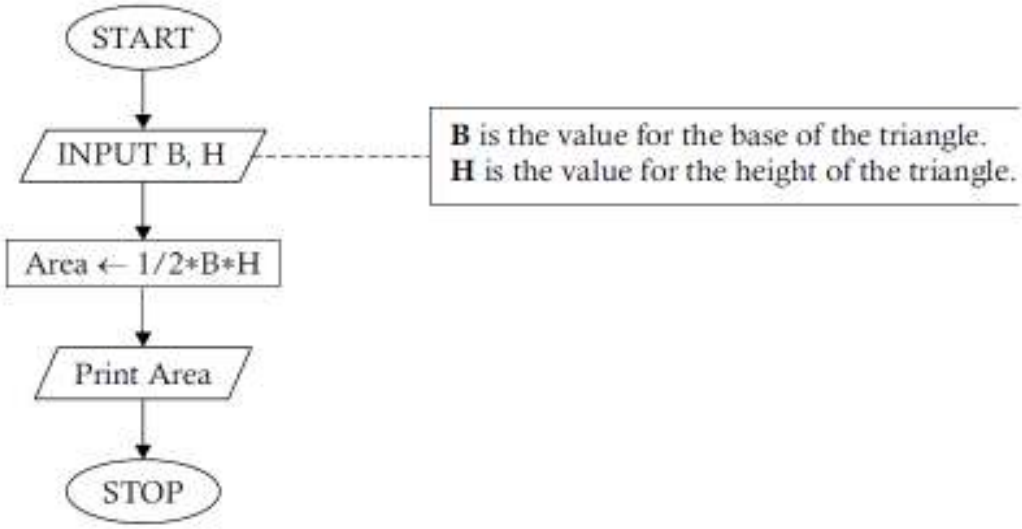
Step 1. INPUT TO H, R
Step 2. COMPUTE W ← H*R
(Store the product of the values in H and R in W)
Step 3. PRINT W
Step 4. STOP

Problem 1.5. *Construct a flowchart to show how to obtain the area of a triangle on the basis of the base and height.*

Task Analysis. We know that the formula to find out the area of a triangle is

$$\text{Area} = \frac{1}{2} \times \text{base} \times \text{height}$$

The inputs required to obtain the area of a triangle are its base and height. We can then put the values in the above formula to obtain the area.

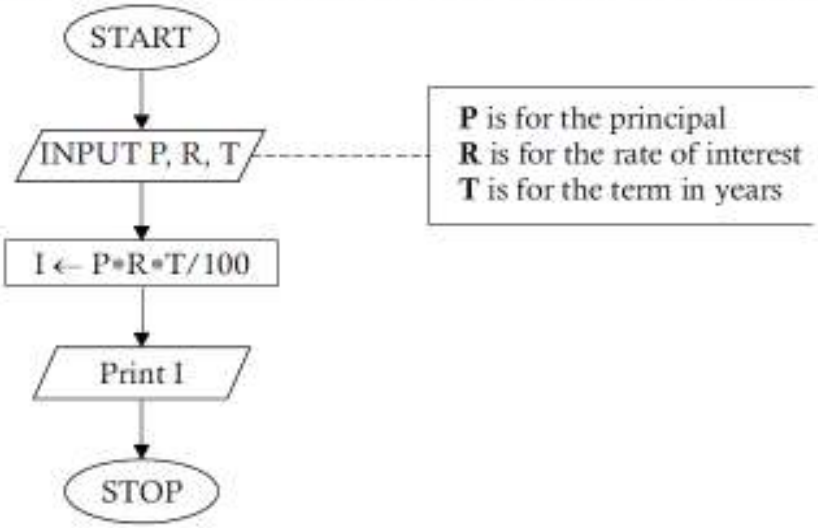


The algorithm corresponding to the above procedure is given below:

Step 1. INPUT TO B, H
(B is for the base and H is for the height of the triangle)
Step 2. COMPUTE AREA $\leftarrow \frac{1}{2}$*B*H
Step 3. PRINT AREA
Step 4. STOP

Problem 1.6. *Develop a flowchart to show the steps in finding the simple interest on a given amount at a given rate of interest.*

Task Analysis. We know that if P is the principal, R is the rate of interest, and T is the term in years, then the simple interest I is given by the formula $I = \frac{P*R*T}{100}$. To determine the simple interest on a given amount, we need the principal amount (P), the rate of interest (R), and the term in years (T). By putting the values in the formula above, we get the desired simple interest.

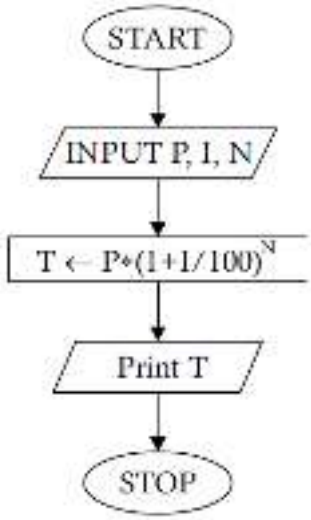


The algorithm corresponding to the above logic is given below:

Step 1. INPUT TO P, R, T
Step 2. COMPUTE I ← P*R*T / 100
Step 3. PRINT I
Step 4. STOP

Problem 1.7. *If P amount of money is invested for N years at an annual rate of interest I, the money grows to an amount T, where T is given by* $T = P(1 + I/100)^N$. *Draw a flowchart to show how T is determined.*

Task Analysis. The solution to this problem is very simple, and it is similar to the preceding one. The inputs required are the values for P, I, and N. The output T can then be obtained by putting the values in the formula.



The algorithm corresponding to Problem 1.7 is given below:

Step 1. INPUT TO P, I, N

Step 2. COMPUTE $T \leftarrow P*\left(1+\frac{I}{100}\right)^N$

Step 3. PRINT T

Step 4. STOP

Problem 1.8. *Construct a flowchart to show how a student's registration number and grades in 3 subjects, m_1, m_2, and m_3, are displayed along with the total average grade.*

Task Analysis. The data supplied as inputs are the registration number and *grades* obtained in three subjects. The registration number contributes nothing to the process of deriving the desired output; it just identifies the person about whom the total grade and the average grade are obtained. The total grade can be obtained by taking the sum of the marks m_1, m_2, and m_3, and the average can be obtained by dividing the total by 3. The steps are illustrated below.

The algorithm corresponding to the above problem is given below:

Step 1. INPUT TO REGN-NO

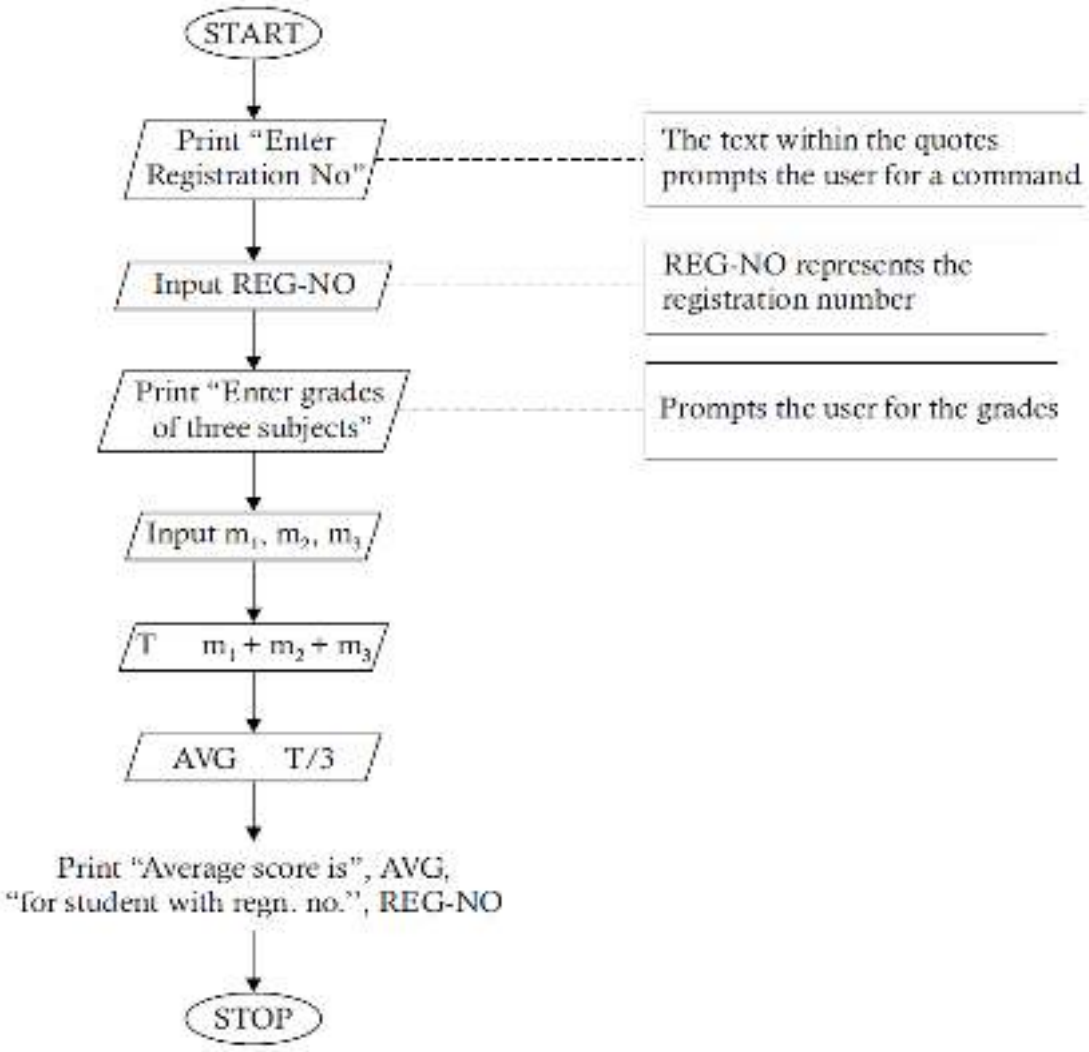
Step 2. INPUT TO M1, M2, M3
(M1, M2, and M3 are for holding the grades in three subjects)

Step 3. COMPUTE T ← M1 + M2 + M3

Step 4. COMPUTE AVG ← T/3

Step 5. PRINT REGN-NO, AVG

Step 6. STOP.

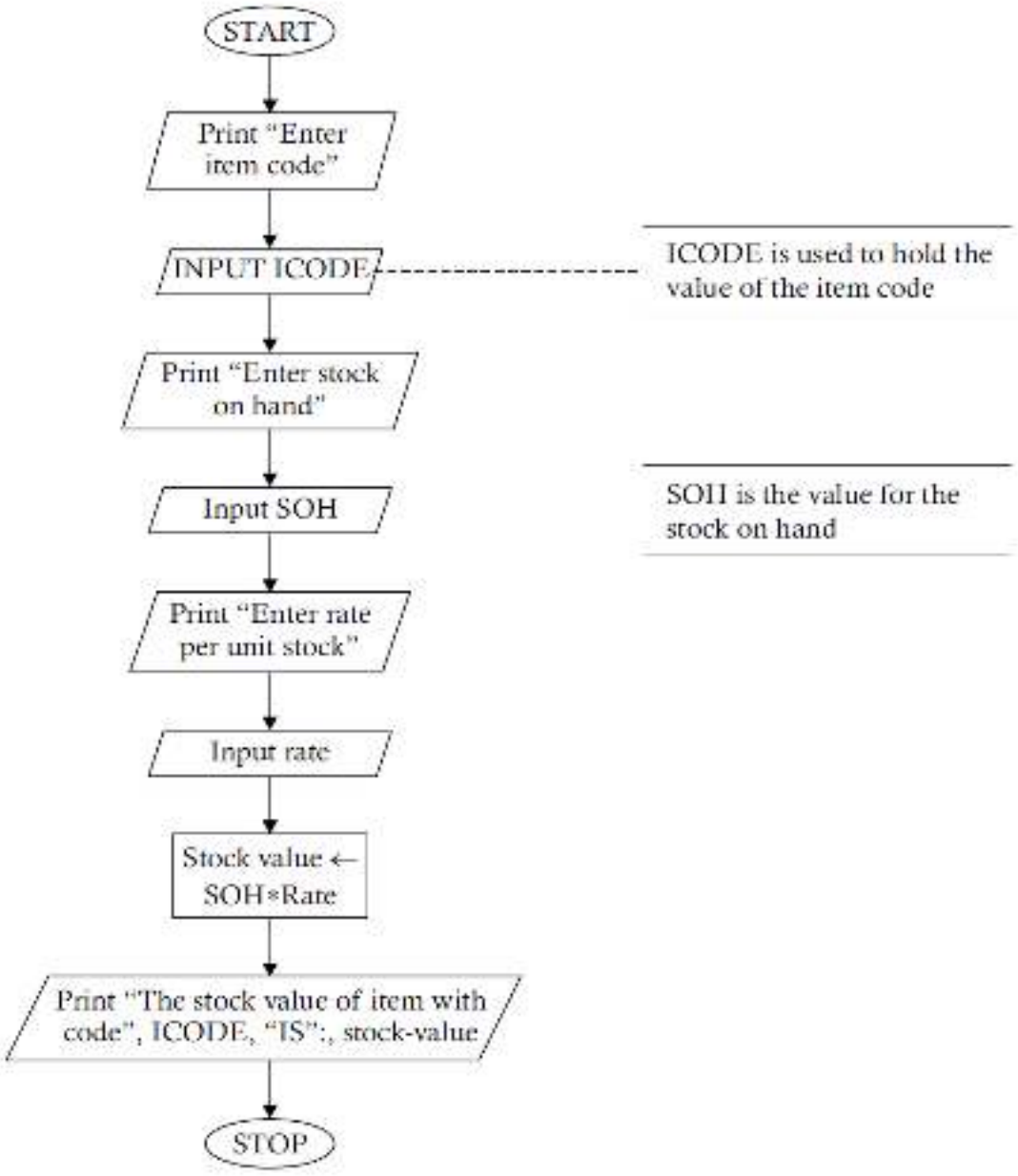


Problem 1.9. *Draw a flowchart to accept the item's code, stock on hand, and the rate per unit of stock in a department store and display the stock value of the store.*

Task Analysis. The inputs required to determine the stock value of the store are the stock on hand and the rate per unit of stock, which are multiplied together to determine the stock value. The item's code is used as the identification data.

The algorithm corresponding to the solution for Problem 1.9 is as follows:

Step 1. INPUT TO I CODE
Step 2. INPUT TO SOH (SOH stands for "stock on hand")
Step 3. INPUT TO RATE



Step 4. COMPUTE STOCK-VALUE ‹ SOH*RATE
Step 5. PRINT STOCK-VALUE, ICODE
Step 6. STOP

EXERCISES

Construct flowcharts to show the steps involved to accomplish the following:

(i) Find the product of two numbers.

(ii) Find the remainder when one number is divided by the other.

(iii) Find the area of a parallelogram.

(iv) Find the area of the four walls of a rectangular room.

(v) Find the area and perimeter of a circular plot.

(vi) Find the area of a triangle based on the length of three sides.

(vii) Find the area and perimeter of a square.

(viii) Find the cost of fencing a rectangle at a given rate.

(ix) Find the surface area of a cone.

(x) Find the volume and surface area of a sphere.

(xi) Convert meters to kilometers.

Algorithm & Flowchart to find Area and Perimeter of Square

L : Side Length of Square
AREA : Area of Square
PERIMETER : Perimeter of Square

Algorithm

Step-1 Start

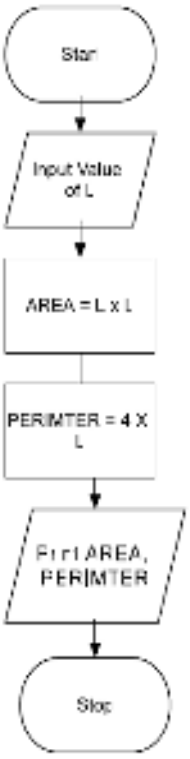
Step-2 Input Side Length of Square say L

Step-3 Area = L x L

Step-4 PERIMETER = 4 x L

Step-5 Display AREA, PERIMETER

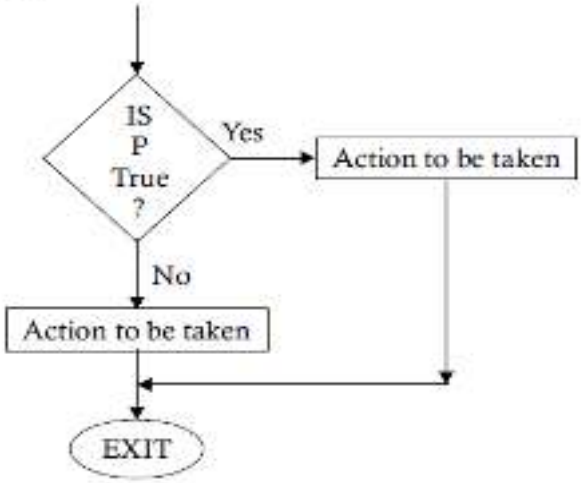
Step-6 Stop



2 PROBLEMS INVOLVING SELECTION

INTRODUCTION

This chapter deals with problems involving decision-making. This process of decision-making is implemented through a logic structure called *selection*. Here a *predicate*, also called a *condition*, is tested to see if it is true or false. If it is true, a course of action is specified for it; if it is found to be false, an alternative course of action is expressed. We can express this process using flowchart notation.



Note that a course of action may involve one or more sequences of operations, and there should be a common meeting point to satisfy the single rule pointed to by the connector containing the word "Exit." A flowchart may contain any number of decision boxes depending on the processing requirements, and the boxes may appear in any sequence depending on the program logic decided. For example, a number of decision boxes may follow one another. The following flowcharts provide an explanation of the logic to clarify this concept.

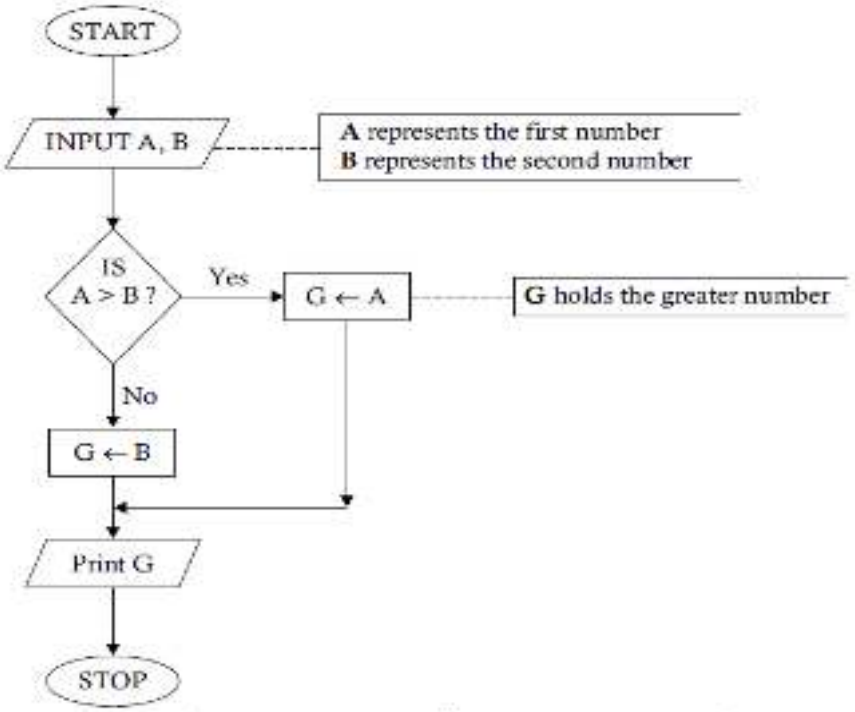
Problem 2.1. *Develop a flowchart to show how the profit or loss for a sale can be obtained.*

Task Analysis. The profit or loss for a sale can be obtained if the cost price and sale price are known. However, there is a need to make a decision here. If the cost price is more than the sale price, then it indicates a loss in the process; otherwise, there will be either zero profit (no profit or a loss) or some profit.

The algorithm corresponding to Problem 2.1 is given below:

Step 1. INPUT TO CP, SP
Step 2. IF CP <= SP

```
THEN
        IF CP = SP
            PRINT "NO PROFIT OR LOSS"
        ELSE
```

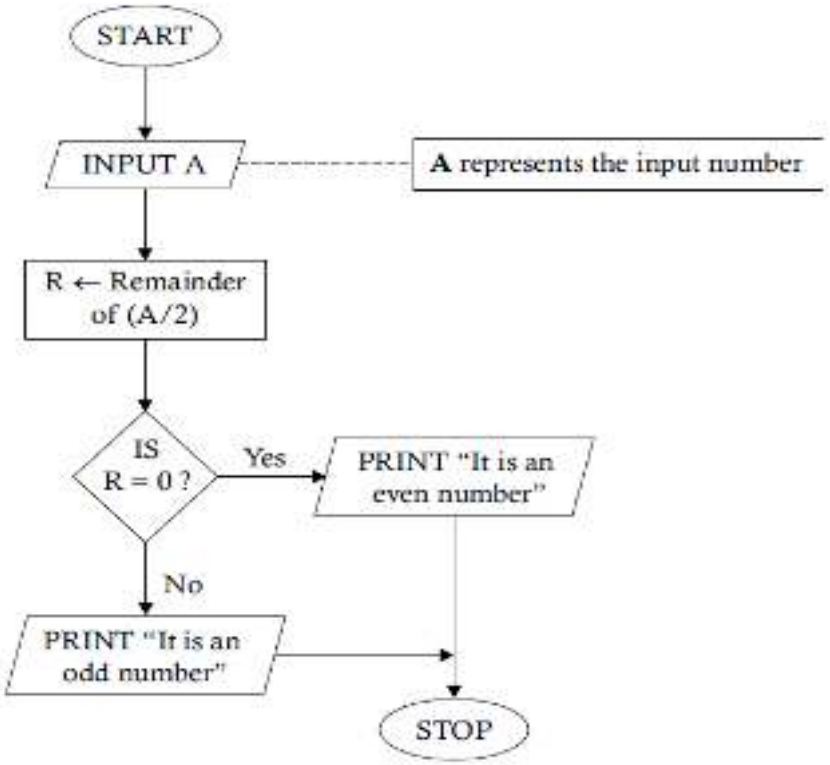


Problem 2.3. *Construct a flowchart to determine whether a given number is even or odd.*

Task Analysis. We know that a number is an even number if it is completely divisible by 2. This means that if we perform integer division upon the given number, then the remainder of the division will be zero. To construct the flowchart, we accept a number as input, obtain the remainder of the integer division by taking it as the divisor, and then check whether the remainder is zero. If it is zero, then our conclusion will be that the number is an even number; otherwise, it will be an odd number.

The algorithm corresponding to Problem 2.3 is shown below:

Step 1. INPUT TO A
Step 2. COMPUTE R ← Remainder of (A/2)
Step 3. IF R = 0
    THEN PRINT "It is an even number."
    ELSE
    PRINT "It is an odd number."
    END-IF
Step 4. STOP



Problem 2.4. *Determine the net payable amount on a sale. The net payable amount consists of the sale price plus sales tax. The sales tax is decided as*

a. *8% of the sale price for national items*
b. *18% of the sale price for foreign items*

Construct a flowchart to show how the net payable amount is determined.

Task Analysis. We need to calculate the sales tax first by taking one of the two given rates. For this purpose, we require two inputs: the sale price of the item under consideration and the origin of the item. Let us assume that we provide "N" or "F" as the input to indicate "national" or "foreign," respectively.

The algorithm corresponding to Problem 2.4 is shown below:

Step 1. INPUT TO SP
Step 2. INPUT TO CHOICE ("N" for national and "F" for foreign)
Step 3. IF CHOICE = "N"
THEN COMPUTE ST ← SP*.08
ELSE
COMPUTE ST ← SP*.18
END-IF
COMPUTE NP ← NP+ST
Step 4. PRINT NP
Step 5. STOP

Problem 2.5. *An equation with the form $ax^2 + bx + c = 0$ is known as a quadratic equation. Draw a flowchart to show how to solve a quadratic equation.*

Task Analysis. The values a, b, and c in the equation represent constant values. So $4x^2 - 17x - 15 = 0$ represents a quadratic equation where $a = 4$, $b = -17$, and $c = -15$. The values of x that satisfy a particular quadratic equation are known as the roots of the equation. The roots may be calculated by substituting the values of a, b, and c into the following two formulas:

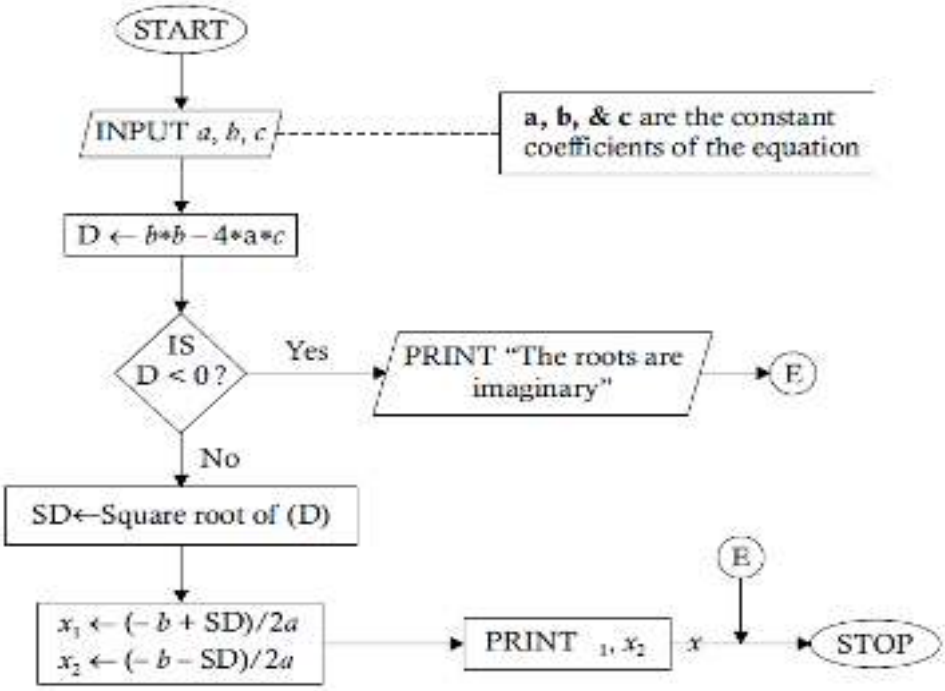
$$x_1 = (-b + \sqrt{b^2 - 4ac}) / 2a$$
$$x_2 = (-b + \sqrt{b^2 - 4ac}) / 2a$$

The expression $b^2 - 4ac$ is called the *determinant* of the equation because it determines the nature of the roots of the equation. If the value of the determinant is less than zero, then the roots of the equation x_1 and x_2, are imaginary (complex) numbers. To solve a quadratic equation, we should allow the user to enter the values for a, b, and c. If the discriminant is less than zero, then a message should be displayed stating that the roots are imaginary; otherwise, the program should proceed to calculate and display the two roots of the equation.

The algorithm corresponding to Problem 2.5 is as follows:

Step 1. INPUT TO A, B, C
Step 2. COMPUTE D ← (B*B – 4*A*C) (Calculate the value of the discriminant) and store in D

Step 3. IF D < 0
    THEN PRINT "'THE ROOTS ARE IMAGINARY"
    ELSE
    COMPUTE SD ← SQUARE-ROOT (D)
    END-IF
Step 4. COMPUTE XI ← (– b + SD)/2*A
Step 5. COMPUTE X2 ← (– b – SD)/2*A
Step 6. PRINT XI, X2
Step 7. STOP



Problem 2.8. *The following rules are used to calculate the bonus for the employees of an organization.* مكافأة(جائزة)

(i) *If the pay is more than $3,000, the bonus amount is fixed, and it is equal to $300.*

(ii) *If the pay is more than $1,600, but less than or equal to $3,000, the bonus will be 10% of the pay subject to a maximum of $240.*

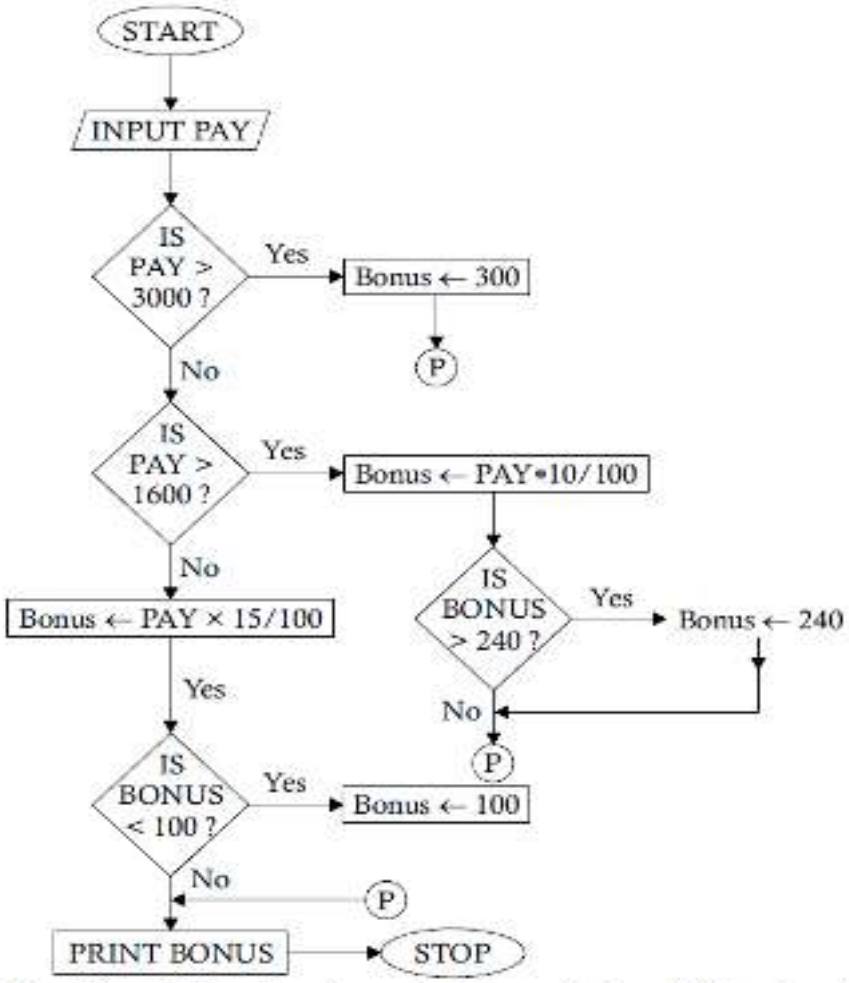
(iii) *If the pay is less than or equal to $1,600, the bonus is 15% of pay, subject to a minimum of $100.*

Task Analysis. The input required here is the pay amount that an employee gets. On the basis of the pay, we can determine the bonus amount. The "subject to maximum" or the "subject to minimum" clause implies that the calculated amount should be compared with the maximum or minimum limit. If it is more than the maximum limit or less than the minimum limit, then the maximum limit or the minimum limit will be treated as the legitimate value.

The algorithm corresponding to Problem 2.8 is given below:

```
Step 1.  INPUT TO PAY
Step 2.  IF PAY > 3000
            THEN BONUS ← 300
         ELSE
            IF PAY > 1600
               THEN BONUS ← PAY* 10/100
                       IF BONUS > 240
                          THEN
                                BONUS ← 240
                       END-IF
               ELSE
```

```
                    BONUS ← PAY* 15/100
                      IF BONUS < 100
                        BONUS ← 100
                      END-IF
                  END-IF
              END-IF
Step 3.   PRINT BONUS
Step 4.   STOP
```



Problem 2.11. *A bookseller offers two rates of commissions. If the price of a book is below $100, the rate of commission is 12% of the price, otherwise, it is 18% of the price. Develop a procedure to determine the discount and the net price of a book.*

Task Analysis. The outputs required are the discount and net price of a book. The only input required for this purpose is price of the book. The rates of the discount are constants (fixed). These rates can be used to develop formulas to calculate the discounts in the two different cases. The calculated discount can then be subtracted from the price of the book to obtain the net price.

(c) 7.5% of sales when sales > = $14,000 in Region A and when sales > = $13,000 in Region B.

(ii) Allow the user to perform a simple task on a calculator on the basis of a given choice as follows:

+, −, ×, /, or % representing the arithmetic operators

A Average of two numbers

X Maximum of two numbers

M Minimum of two numbers

S Square of two numbers

Q Quit

(iii) An electricity board charges the following rates to domestic users to discourage large consumption of energy:

for the first 100 units—$.85 per unit

for the next 200 units—$1.45 per unit

Beyond 300 units—$1.85 per unit

All users are charged a minimum of $ 500.00. If the total cost is more than $ 2,500.00, then an additional surcharge of 3% of the total cost is added to the total cost to determine the final bill.

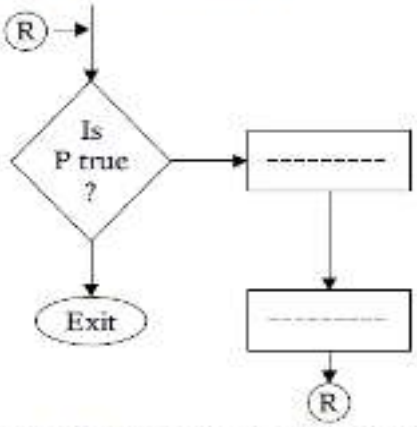
(iv) To determine and print the minimum number of currency notes of the denominations: $1, $5, $10, $20, $50, $100, $500 and $1000 required to pay any given amount.

3 PROBLEMS INVOLVING LOOPING

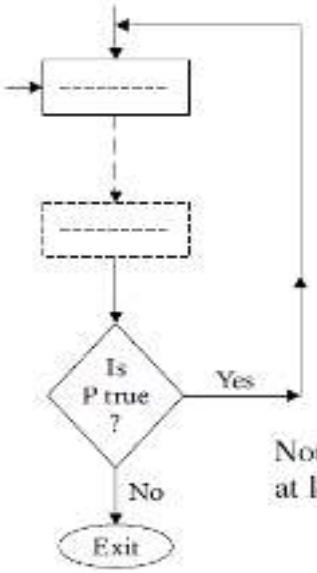
INTRODUCTION

The term *iteration* means repetition. Sometimes, a procedure should be executed repeatedly. All procedures should be built so that they can be repeated as many times as needed. We should not develop procedures to execute only once. Otherwise, calculators could be sufficient to obtain the results. An iterative logic structure is also known as a *loop*. *Looping* means repeating a set of operations to obtain a result repeatedly.

An iteration may be implemented in two ways: a pre-test iteration and post-test iteration. In case of a *pre-test iteration*, a predicate is tested to decide whether a set of operations is to be performed or not. If the condition implied by the predicate is true, then the desired operations are performed. If it is false, then the iteration is terminated. This is shown in the following diagram.

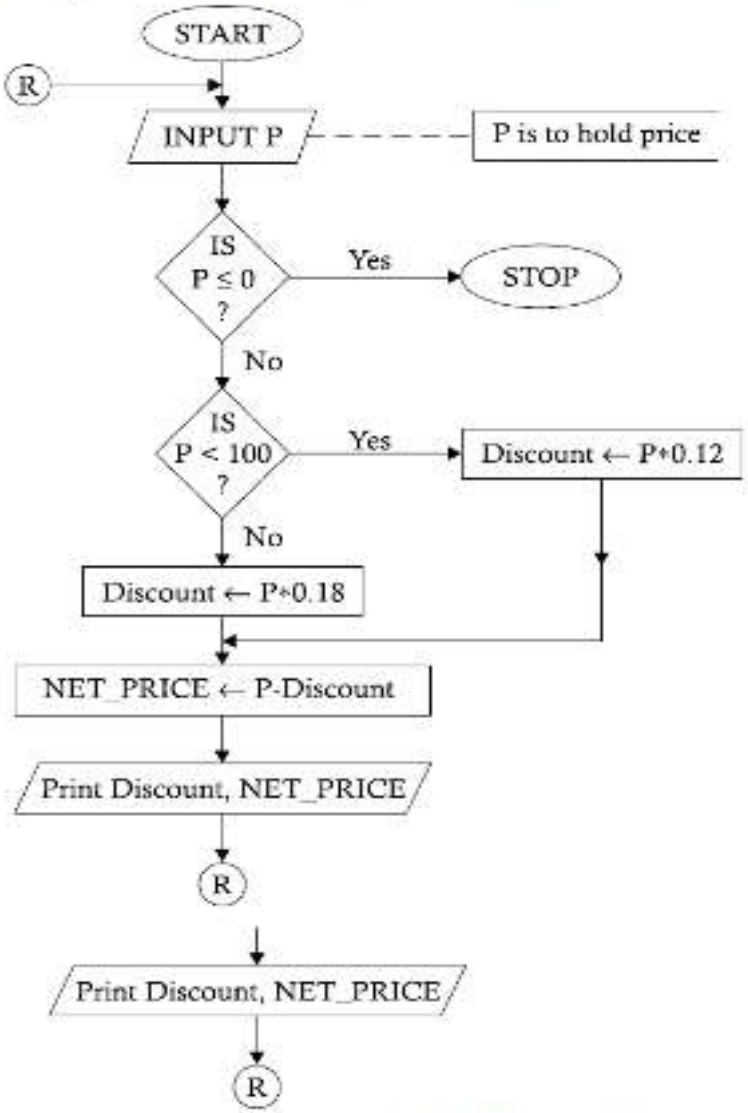


For a post-test iteration, the predicate is tested after performing a set of operations once to decide whether to repeat the set of operations or to terminate the repetition. If the condition happens to be true, then the set of operations is repeated; otherwise, it is not repeated. The diagrammatic structure of this logic is as follows.



Note that the operations in the loop must be performed at least once in the case of a post-test iteration.

The concept of looping is demonstrated in the following flowchart. Of course, there should be a condition for normal termination. Let us assume that the repetitive task of calculating the discounts and net prices is terminated when we provide negative or zero as the price for the input. Such absurd values are justified for the termination of loops so that the procedure can remain valid for any possible value of the price. We usually use out-connectors and in-connectors with the same label to demonstrate the end point and start point of a loop. These are shown in the flowchart of Problem 2.11.



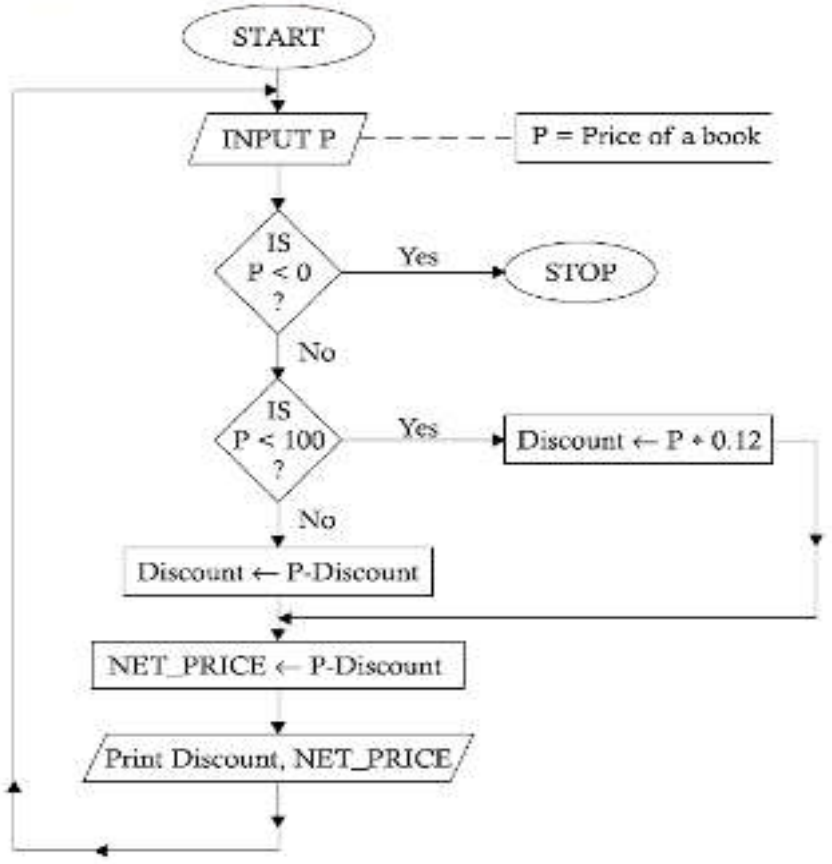
The algorithm corresponding to the flowchart is below:

Step 1. REPEAT STEPS 2 THROUGH 6 (Start Loop)
Step 2. INPUT TO P
Step 3. IF P ≤ 0 THEN EXIT (Stop Repetition, *i.e.*, transfer the control to STOP).

Step 4. If P < 100
THEN COMPUTE D ← P*0.12
ELSE COMPUTE D ← P*0.18
END-IF
Step 5. COMPUTE NET_PRICE ← P – D
Step 6. PRINT D, NET_PRICE (End of loop)
Step 7. STOP

Note that the out-connector (R) shows the end point of the loop and the in-connector. (R)→ shows the start point of the loop. The operations starting from the point of the accepting the input price up to the points of printing the output discount and net price are within the loop. It could have been demonstrated without using connectors.

However, we prefer the first flowchart to the following one, because if the flowchart cannot be accommodated on a single page (or in a continuous structure on a single page), it would be difficult or impossible difficult to connect the start point and the end point.



Problem 3.1. *The salesmen of a sales firm are given a commission on sales achieved, using the following rules:*

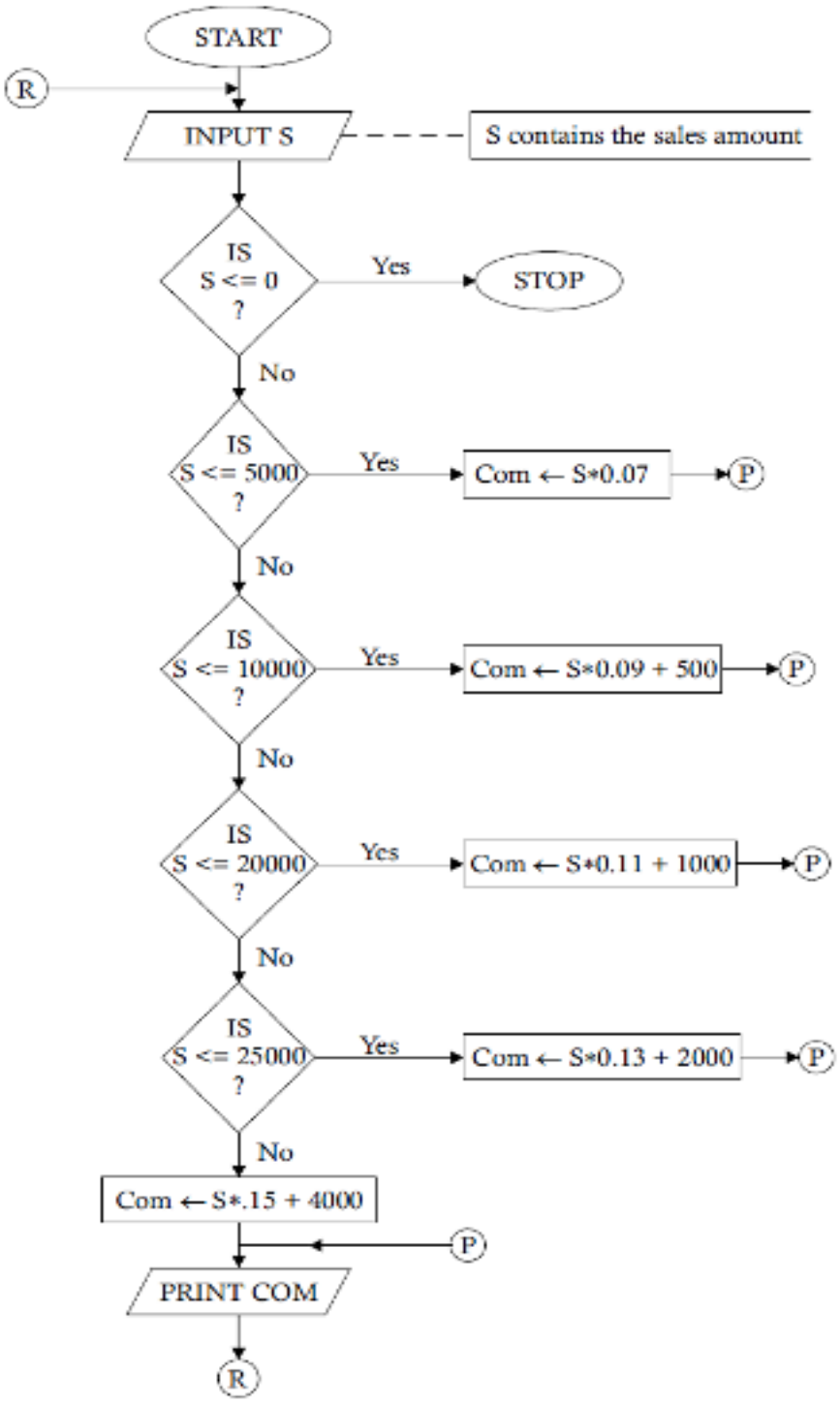
Sales	*Rate of commission*
<= 5,000	*7% of sales*
> 5,000 but <= 10,000	*9% of sales + $500*
> 10,000 but <= 20,000	*11% of sales + $1,000*
> 20,000 but <= 25,000	*13% of sales + $2,000*
> 25,000	*15% of sales + $4,000*

Devise a procedure to calculate the commission of the salesmen.

Task Analysis. The output required is the commission earned by a salesman. The only input required is the amount of the sale. A number of decision-making steps are involved, and the process is likely to be repeated a number of times. Let us assume that the process can be terminated when the amount of the sale is zero or negative. The procedure is illustrated in the following flowchart.

The algorithm corresponding to Problem 3.1 is given below:

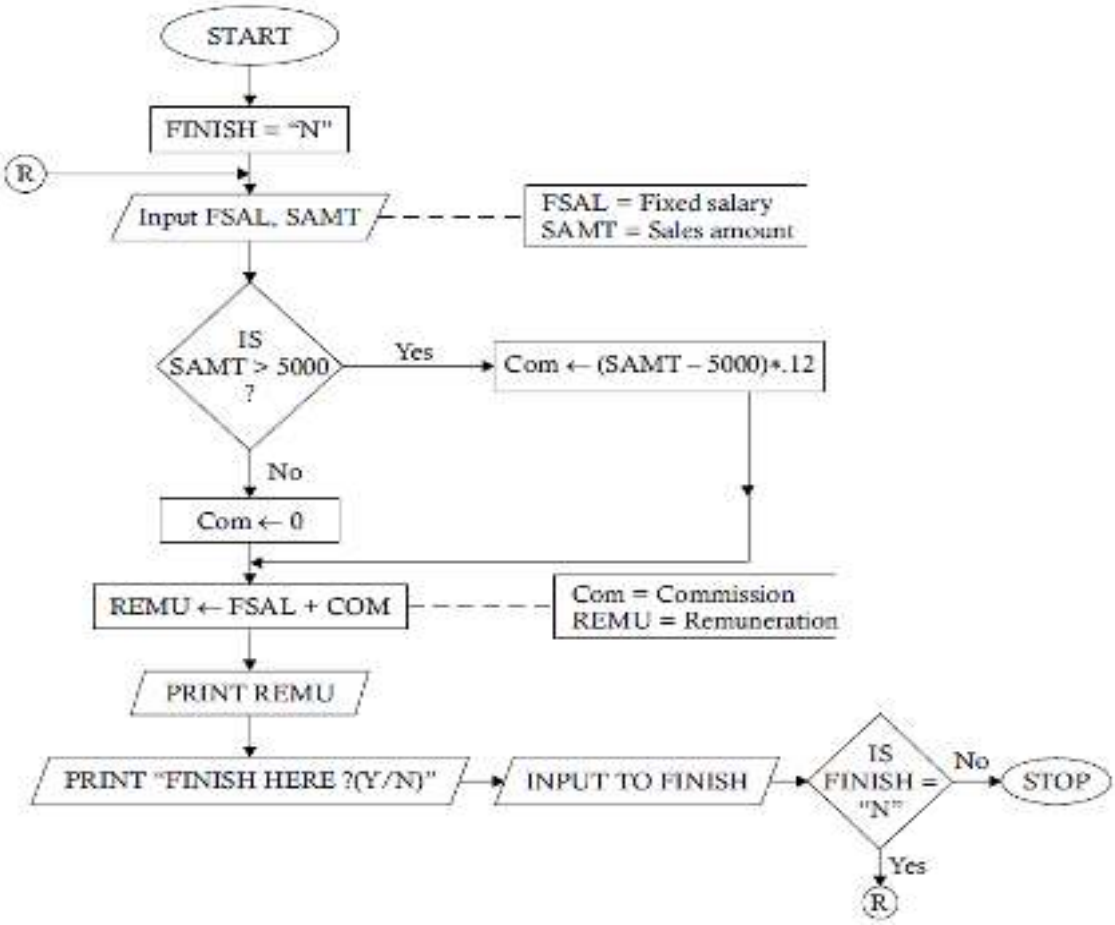
```
Step 1.  REPEAT STEPS 2 THROUGH 5
Step 2.  INPUT TO S
         (Accept sales amount in S)
Step 3.  If S <= 0
           THEN EXIT
         END-IF
Step 4.  IF S <= 5000
           THEN COMPUTE COM ← S * .07
         ELSE
           IF S <= 10000
             THEN COMPUTE COM ← S * .09 + 500
           ELSE
             IF S <= 20000
               THEN COMPUTE COM ← S * 0.11 + 1000
             ELSE
               IF S <= 25000
                 THEN COMPUTE COM ← S * 0.13 + 2000
               ELSE
                 COMPUTE COM ← S * 0.15 + 4000
               END-IF
             END-IF
           END-IF
         END-IF
Step 5.  PRINT COM
Step 6.  STOP
```


START
R
INPUT S
S contains the sales amount
IS S <= 0 ?
Yes
STOP
No
IS S <= 5000 ?
Yes
Com ← S*0.07
P
No
IS S <= 10000 ?
Yes
Com ← S*0.09 + 500
P
No
IS S <= 20000 ?
Yes
Com ← S*0.11 + 1000
P
No
IS S <= 25000 ?
Yes
Com ← S*0.13 + 2000
P
No
Com ← S*.15 + 4000
P
PRINT COM
R

Problem 3.2. *A sales organization offers a fixed salary and a percentage of sales as a commission to determine the monthly remuneration of an employee under the following conditions.*

If the sales amount of an employee exceeds $5,000, then the commission is 12% of the sales that exceed $5,000; otherwise, it is nil. Draw a flowchart to show how the remuneration of an employee is decided.

Task Analysis. The remuneration of an employee consists of two parts: a fixed salary part and a commission part that depends on the sales amount. We use the fixed salary part and the sales amount as input to determine the commission and hence, the remuneration.



The algorithm corresponding to Problem 3.2 is given below:

Step 1. FINISH ← "N"
Step 2. REPEAT STEPS 3 THROUGH 9 WHILE FINISH = "N"
Step 3. INPUT TO FSAL, SAMT
Step 4. IF SAMT > 5000
THEN COMPUTE COM ← (SAMT – 5000) * .12
ELSE

```
            COM ← 0
          END-IF
Step 5.   COMPUTE REMU ← FSAL + COM
Step 6.   PRINT "REMUNERATION IS", REMU
Step 7.   PRINT "FINISH (Y/N)?"
Step 8.   INPUT TO FINISH
Step 9.   IF FINISH = "Y"
            THEN EXIT
          END-IF
Step 10.  STOP
```

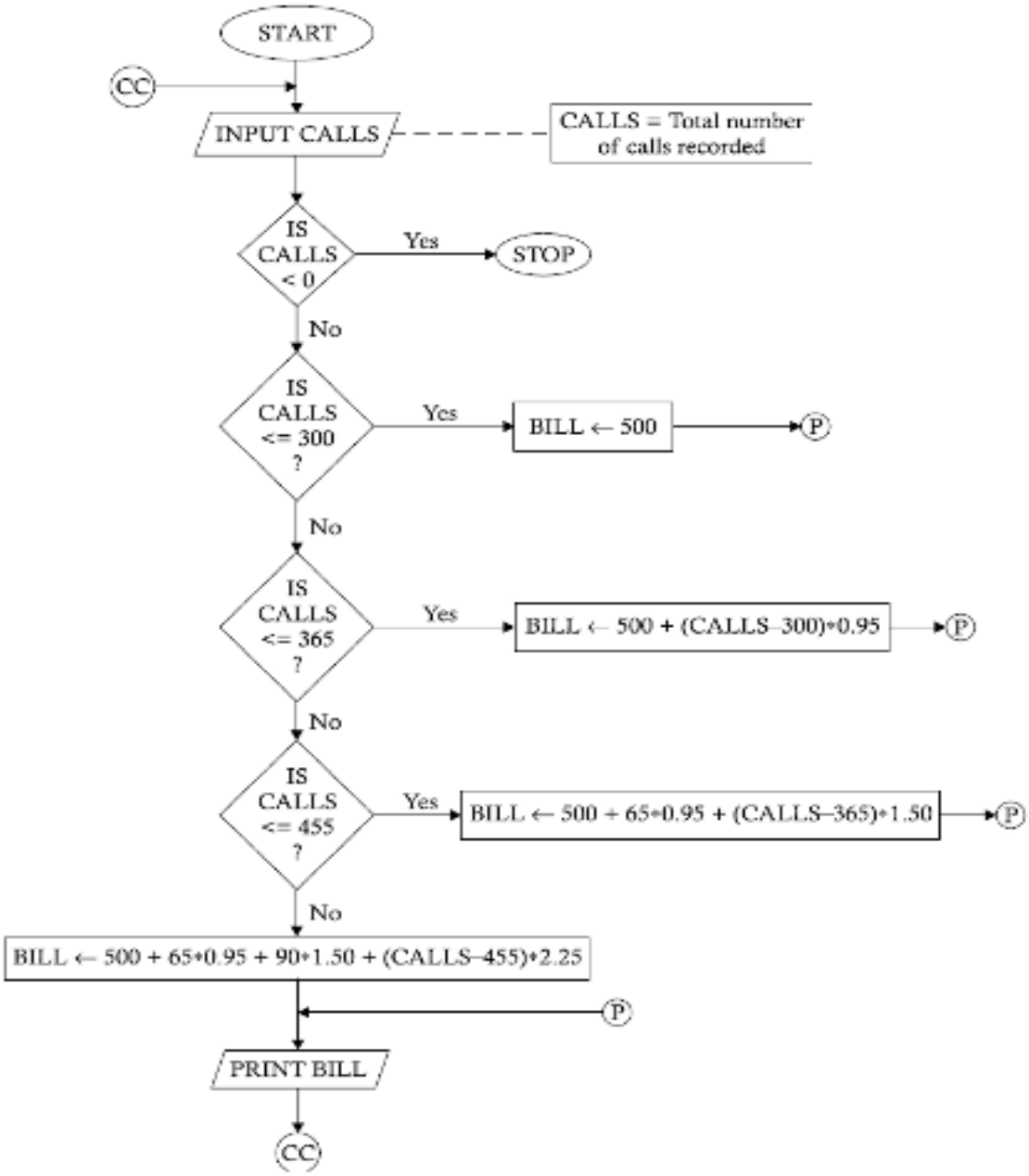
Problem 3.3. *A labor contractor pays the workers at the end of each week according to the rules given below:*

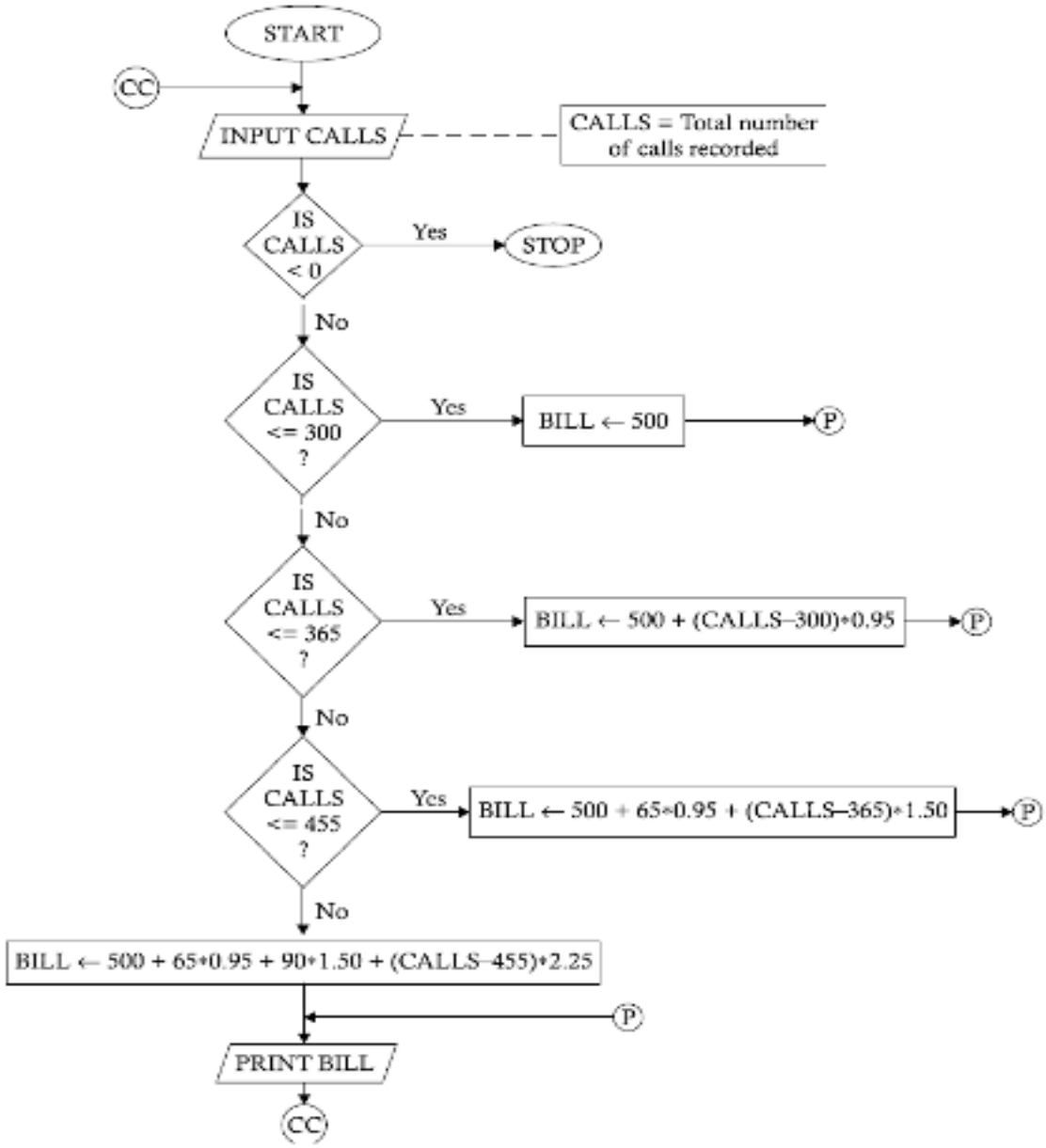
For the first 35 hours of work, the rate of pay is $15 per hour; for the next 25 hours, the rate of pay is $18 per hour; for the rest, the rate of pay is $26 per hour. No worker is allowed to work for more than 80 hours in a week. Develop a flowchart to show how the wages of the workers can be calculated on the basis of calid inputs.

Task Analysis. The input required is the total number of hours worked. The rates of payment depend on the different numbers of hours worked. The total hours worked may be considered valid if the number lies in the range of 0 through 80. Our procedure for evaluating the wage consists of the (*i*) validation of the hours worked, (*ii*) identifying the category to which the hours worked pertain, and then (*iii*) applying different rates to calculate the wage. The procedure is shown within a loop, and it is terminated when zero or a negative value is given as the input against hours worked.

The algorithm corresponding to Problem 3.3 is given below:

```
Step 1.   REPEAT STEPS 2 THROUGH 6
Step 2.   INPUT TO TH
Step 3.   IF TH <= 0
            THEN EXIT
          END-IF
Step 4.   IF TH > 80
            THEN PRINT "INVALID HOURS"
            CONTINUE
          END-IF
Step 5.   IF TH <= 35
            THEN COMPUTE WAGE ← TH*15
          ELSE
            IF TH <= 60
              THEN COMPUTE WAGE ← 35*15 + (TH–35)*18
          ELSE
            COMPUTE WAGE ← 35*15 + 25*18 + (TH–60)*25
          END-IF
        END-IF
Step 6.   PRINT "WAGE IS", WAGE
Step 7.   STOP
```

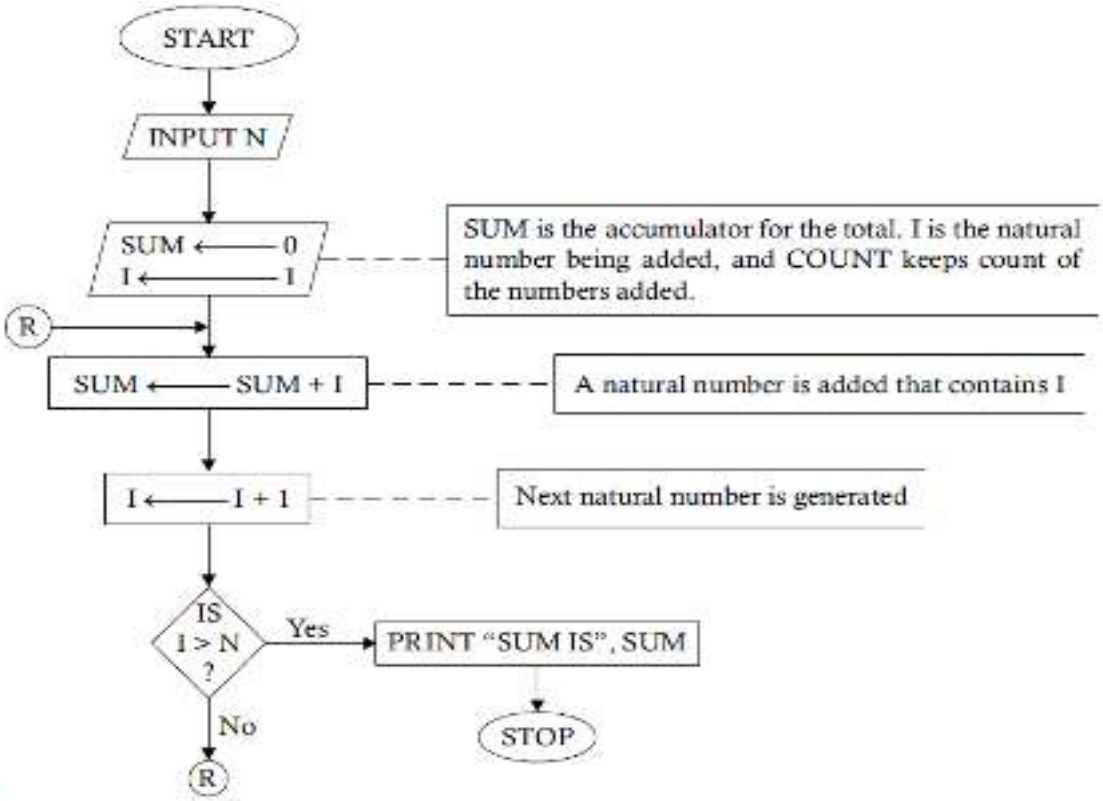

START
CC
INPUT CALLS
CALLS = Total number of calls recorded
IS CALLS < 0
Yes
STOP
No
IS CALLS <= 300 ?
Yes
BILL ← 500
P
No
IS CALLS <= 365 ?
Yes
BILL ← 500 + (CALLS–300)*0.95
P
No
IS CALLS <= 455 ?
Yes
BILL ← 500 + 65*0.95 + (CALLS–365)*1.50
P
No
BILL ← 500 + 65*0.95 + 90*1.50 + (CALLS–455)*2.25
P
PRINT BILL
CC



Problem 3.6. *Devise a procedure to find the sum of first n natural numbers, where n is any given integer, without using a formula.*

The algorithm corresponding to Problem 3.6 is shown below:

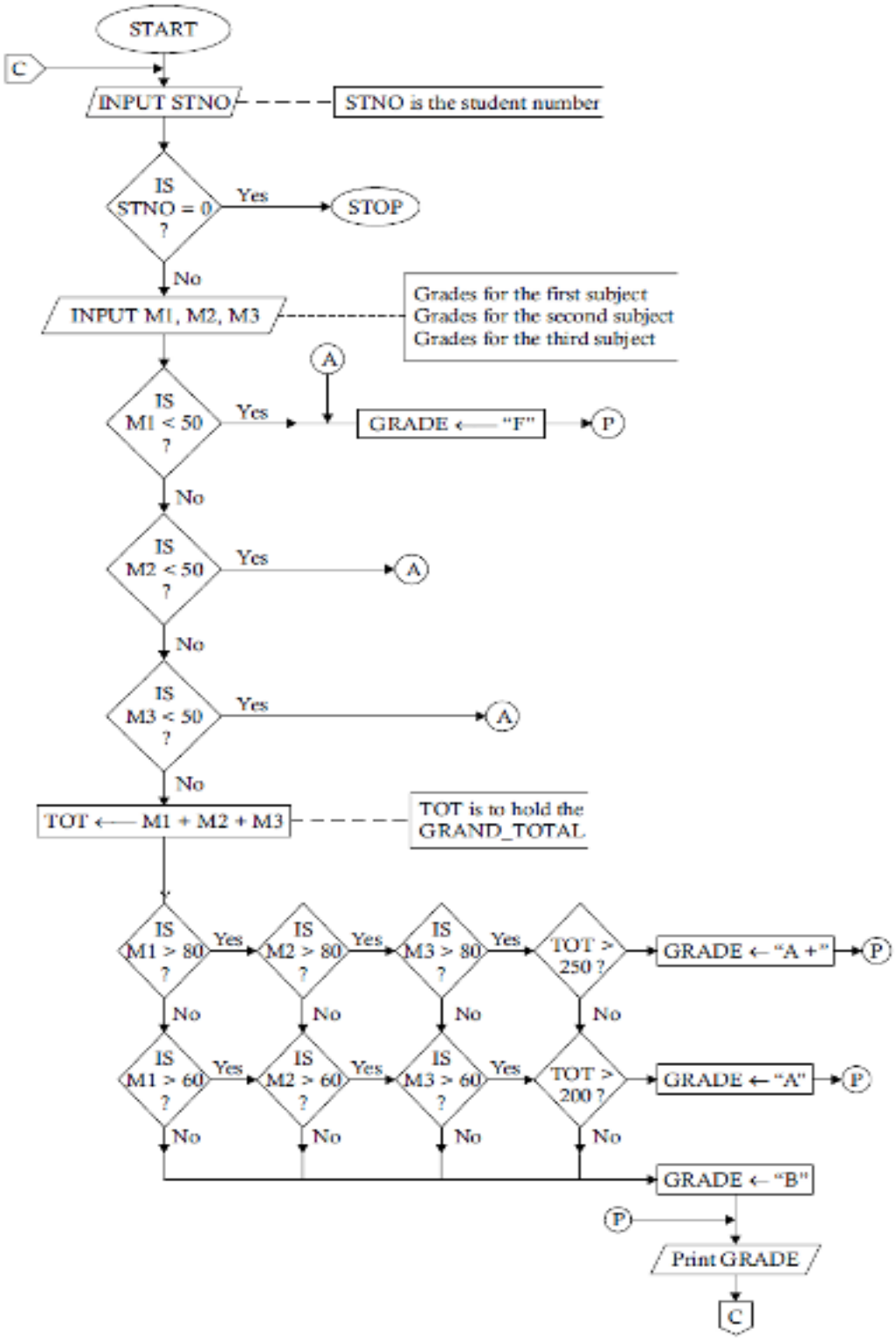
Step 1. INPUT "ENTER NUMBER OF TERMS TO ADD" TO N
Step 2. SUM ← 0 [INITIALIZATION]
Step 3. I ← 1 [INITIALIZATION]
Step 4. REPEAT STEPS 5 THROUCH 6 WHILE I <= N.
Step 5. COMPUTE SUM ← SUM + I
Step 6. COMPUTE I ← I + 1
Step 7. PRINT "THE SUM IS", SUM
Step 8. STOP



Problem 3.13. *Draw a flowchart for the following problem to determine the grade. There are 3 tests for 3 different subjects. On the basis of grades in the three subjects, M1, M2, and M3, a grade is awarded to each student as per the following rules:*

a. *If the score in each subject is more than 80, and the total is more than 250, the grade is A +*
b. *If the score in each subject is more than 60, and the total is more than 200, the grade is A*
c. *If the score in any one or more subjects is less than 50, the grade is F*
d. *In all other cases, the grade is B.*

Task Analysis. This problem involves the task of nested decision-making. The inputs required are the grades for the three subjects. In addition, a student identification number may be accepted as input. The procedure will include calculation of the total grades obtained by a student and then comparison of the individual grades and the total grades according to the rules of gradation to determine the grade.


START
C
INPUT STNO
STNO is the student number
IS STNO = 0 ?
Yes
STOP
No
INPUT M1, M2, M3
Grades for the first subject
Grades for the second subject
Grades for the third subject
A
IS M1 < 50 ?
Yes
GRADE ← "F"
P
No
IS M2 < 50 ?
Yes
A
No
IS M3 < 50 ?
Yes
A
No
TOT ← M1 + M2 + M3
TOT is to hold the GRAND_TOTAL
IS M1 > 80 ?
Yes
IS M2 > 80 ?
Yes
IS M3 > 80 ?
Yes
TOT > 250 ?
GRADE ← "A +"
P
No
No
No
No
IS M1 > 60 ?
Yes
IS M2 > 60 ?
Yes
IS M3 > 60 ?
Yes
TOT > 200 ?
GRADE ← "A"
P
No
No
No
No
GRADE ← "B"
P
Print GRADE
C

The solution of Problem 3.13 is shown in the following algorithm:

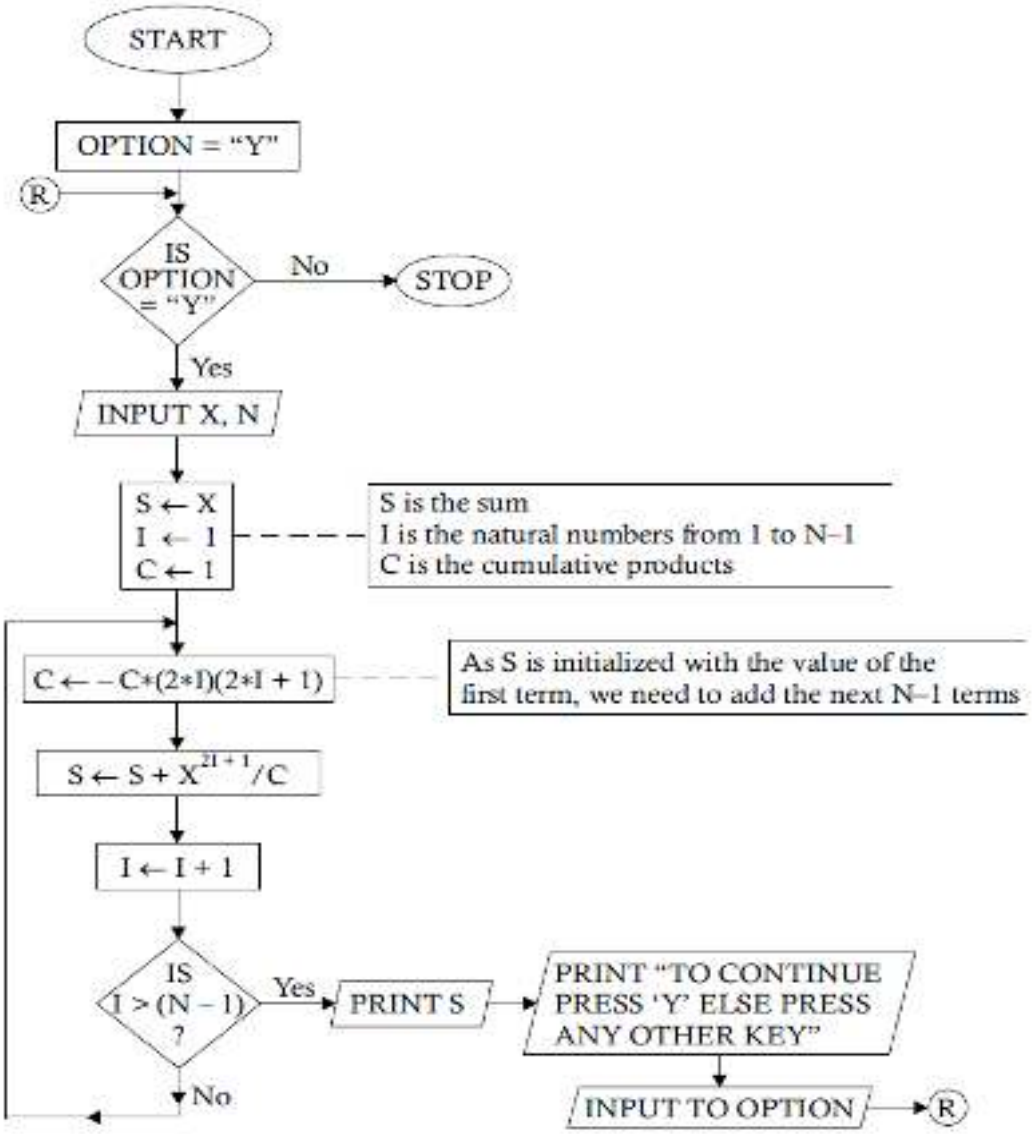
Step 1. FOR EACH STNO DO
Step 2. INPUT TO STNO
Step 3. IF STNO = 0 THEN EXIT
END-IF
Step 4. INPUT TO M1, M2, M3
Step 5. IF M1 < 50 OR M2 < 50 OR M3 < 50
THEN GRADE ← "F"
END-IF
Step 6. COMPUTE TOT ← M1 + M2 + M3
Step 7. IF M1 > 80 AND M2 > 80 AND M3 > 80 AND TOT > 250
THEN GRADE ← "A+"
ELSE IF M1 > 60 AND M2 > 60 AND M3 > 60 AND TOT > 200
THEN GRADE ← "A"
ELSE
GRADE ← "B"
END-IF
END-IF
Step 8. PRINT GRADE
Step 9. END-FOR
Step 10. STOP

Problem 3.33. *Develop a flowchart to show how to evaluate the following series:*

$$X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \ldots\ldots\ldots \text{ up to } N \text{ terms.}$$

The algorithm leading to the solution of Problem 3.33 is given below:

Step 1. [INITIALIZE LOOP VARIABLE]
OPTION ← "Y"
Step 2. REPEAT STEPS 3 THROUGH 8 WHILE OPTION = "Y"
Step 3. PRINT "ENTER VALUE FOR X"
Step 4. ACCEPT X
Step 5. PRINT "ENTER THE NUMBER OF TERMS TO BE ADDED"
Step 6. ACCEPT N
Step 7. [INITIALIZE SUM ACCUMULATOR, LOOP VARIABLE, & COEFFICIENT VARIABLE]
S ← X, I ← 1, C ← 1
Step 8. WHILE I <= N DO
(*a*) COMPUTE C ← − C*(2*I)*(2*I + 1)
[Obtain the $(I + 1)^{th}$ COEFFICIENT]
(*b*) COMPUTE S ← S + $X^{(2*I + 1)}$/C
[Obtain sum of the $(I + 1)^{th}$ term]
(*c*) COMPUTE I ← I + 1
[INCREMENT LOOP VARIABLE]
Step 9. PRINT "THE SUM IS", S
Step 10. STOP


START
OPTION = "Y"
R
IS OPTION = "Y"
No
STOP
Yes
INPUT X, N
S ← X
I ← 1
C ← 1
S is the sum
I is the natural numbers from 1 to N–1
C is the cumulative products
C ← – C*(2*I)(2*I + 1)
As S is initialized with the value of the first term, we need to add the next N–1 terms
S ← S + X^(2I + 1)/C
I ← I + 1
IS I > (N – 1) ?
Yes
PRINT S
PRINT "TO CONTINUE PRESS 'Y' ELSE PRESS ANY OTHER KEY"
INPUT TO OPTION
R
No

4 PROBLEMS INVOLVING ARRAYS