SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES,
CHENNAI-602105
SAVEETHA SCHOOL OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course Code : CSA5779

Cours Name : Fundamentals of Computing for Data Analysis

List of Experiments

1. Generation of number series 1, 2, 3, 4,…..n
2. Generation of even number series 2, 4, 6, …..n
3. Generation of ODD number series 1, 3, 5, …..n
4. Generation of Fibonacci series 0, 1, 1, 2, 3, 5, 8, …..n
5. Summing up series 1 + 2 + 3 + 4….. +n
6. Summing up Even Number series
7. Summing up Odd Number series
8. Summing up 1 – 2 + 3 – 4 + 5…. N
9. Summing up $1^2 + 2^2 + 3^2 + 4^2$….. +n
10. Summing up $2^2 + 4^2 + 6^2 + 8^2$ + ….. n^2
11. Summing up $1^1+2^2+3^3+4^4+$ …. n^n
12. Summing up squares of Odd numbers
13. Summing up cubes of n numbers
14. Product series (Factorial of a given number)
15. Finding given number is Armstrong or not
16. Summing up any n numbers and finding average
17. Printing digits of an integer number
18. Summing up the digits of an integer number
19. Revering the digits of an integer number
20. Finding whether the given integer is odd or even
21. Finding the given integer is positive or negative
22. Swapping two numbers with a temporary variable
23. Swapping two numbers without a temporary variable
24. Swap 3 numbers a to b, b to c and c to a
25. Finding the biggest out of 2 integer numbers
26. Finding the biggest out of n integers
27. Sine series [$\sin(x) = x - x^3/3! + x^5/5! - x^7/7!$ . . . . .]
28. Cos series [$\cos(x) = 1 - x^2/2! + x^4/4! - x^6/6!$ . . . . .]
29. Exponential series [$e^{-1} = 1 - x/1! + x^2/2! - x^3/3! + x^4/4!$ . . . . .]
30. Linear Search

31. Calculate the water bill given the cubic feet of water used for Eureka Water Company, which charges the homeowner one of the following:

a. A flat rate of $15.00 for usage up to and including 1000 cubic feet.
b. $0.0175 per cubic foot for usage over 1000 cubic feet and up to and including 2000 cubic feet.
c. $0.02 per cubic foot for usage over 2000 cubic feet and up to and including 3000 cubic feet.
d. A flat rate of $70.00 for usage over 3000 cubic feet.

Write the algorithm, draw the flowchart and write pseudocode to test the above problem.

32. A company that issues check-cashing cards uses an algorithm to create card numbers. The algorithm adds the digits of a four-digit number, and then adds a fifth digit of 0 or 1 to make the sum of the digits even. The last digit in the number is called the *check digit.* Complete the seven problem-solving steps to develop a solution that accepts a four-digit number into one variable, adds the check digit, and prints the original number and the new number. Test your algorithm, flowchart and pseudocode with the following data: Original (47371) and 4631 (46310).

Hint: You may use any or all of these functions and the principle of concatenation of strings.

Integer(X)—Integer function
String(X)—Numeric to string
Value(A)—String to numeric

Note: The *Integer(X)* gives the whole number value of the real number *X.* When *X* is 546.43, the *Integer(X)* is 546; when *X* is 23.899 the *Integer(X)* is 23. The *String(X)* and *Value(A)* are conversion functions. The resultant of the function *String(X)* is the string value of the numeric *X.* The resultant of *Value(A)* is the numeric value of the string *A.* Concatenation is the combining of strings by placing the first string in front of the second one. For example, the resultant of would be "45."

33. An admission charge for The Little Rep Theatre varies according to the age of the person. Develop a solution to print the ticket charge given the age of the person.

The charges are as follows:

a. Over 55: $10.00
b. 21–54: $15.00
c. 13–20: $10.00
d. 3–12: $5.00
e. Under 3: Free

34. A hotel has a pricing policy as follows:
    a. 2 people: $85
    b. 3 people: $90
    c. 4 people: $95
    d. Additional people: $6 per person
    If the customer is staying on company business, there is a 20% discount. If the customer is over 60 years of age, there is a 15% discount. A customer does not receive both discounts. Given the above data, print the cost of the room.

35. A student wants to know his grade point average for the semester. The grades are given in letter grades with numeric equivalents. Develop a solution to calculate a grade point average given the letter grades. (Remember, the grade point average is figured per unit of credit, not per course.) An A = 4.0, B = 3.0, C = 2.0, D = 1.0, F = 0.0. Write the algorithm to test the solution with the following data and draw flowchart and write pseudocode:

History	B	3 units
Economics	A	3 units
PE	A	1 unit
Chemistry	C	4 units
Art	B	3 units

    (*Hint:* Use a trip value to stop the processing of the loop and a case structure to find the grade points.)

36. Mr. Johnson would like to know how many As, Bs, Cs, Ds, and Fs his students received on a test. He has 200 students who took the test. He would like to enter the student number and the number grade for the test for each student. Develop the solution to print out each student's student number, number grade, letter grade, and the total number of As, Bs, Cs, Ds, and Fs. His grading scale is as follows: 90–100 is an A, 78–89 is a B, 65–77 is a C, 50–64 is a D, and below 50 is an F. Write the algorithm, draw the flowchart and write pseudocode to test the above problem.

37. John Smith is a new car salesperson. Write the algorithm to calculate the total cost of a car given the following. In Addition, write the pseudocode and draw the flowchart:

    initial price of the car
    0 to 10 accessories (the computer would select the price according to the accessory)
    sales tax

38. The Last Stop Boutique is having a five-day sale. Each day, starting on Monday, the price will drop 10% of the previous day's price. For example, if the original price of a product is $20.00, the sale price on Monday would be $18.00 (10% less than the original price). On Tuesday the sale price would be $16.20 (10% less than Monday). On Wednesday the sale price would be $14.58; on Thursday the sale price would be $13.12; and on Friday the sale price would be $11.81. Develop a solution that will calculate the price of an item for each of the five days, given the original price. Write the algorithm, flowchart and pseudocode to test the solution.

39. Mary Smith, a student, has borrowed $3,000 to help pay her college expenses. After setting up a budget, $85 was the maximum monthly payment she could afford to make on the loan. Develop a solution to calculate and print the interest, the principal, and the balance on the loan per month. Other information she would like to know is the number of years and months it will take to pay the loan back and the total interest she will pay during that period. The interest rate is 1% per month on the unpaid balance. Write the algorithm, flowchart and pseudocode to test the solution.

40. Write a solution (algorithm, flowchart and pseudocode) to find the average miles per gallon on a car after six fillups at a gas station. Additional data kept included the number of gallons of gas at each fillup, the starting odometer reading, and the odometer reading at each fillup.

41. Develop a solution (algorithm, flowchart and pseudocode) to calculate a student's grade average for one semester. The letter grades should be entered and the grade average printed out. An A is equivalent to 4 grade points, a B is 3 grade points; a C is 2 grade points, a D is 1 grade point, and an F is zero grade points.

42. Mr. Jones always gives True/False tests to his class. His tests always have 20 questions. The maximum class size is 35. He needs a program that will calculate the students' rades based on the best score.

    Grade
    A will range from the best score, to the best score minus 2.
    B will range from the best score minus 3, to the best score minus 4.
    C will range from the best score minus 5, to the best score minus 6.
    D will range from the best score minus 7, to the best score minus 8.
    F will be anything below the best score minus 8.

    Each student's ID and test answers will be entered. The output will be each student's ID, number correct, and grade, along with the single highest score for the class. Develop the algorithm, draw the flowchart and write the pseudocode for Mr. Jones's problem. Use four one-dimensional arrays—one for the correct scores and the other three for the needed output.

43. A restaurant manager wants to know how many employees are needed at the restaurant each hour of the day. The minimum number of employees needed at any hour is 3. After that, one additional employee is required for each 20 customers. The restaurant is open 24 hours a day. The manager has counted the number of customers each hour for 14 days. The manager will use the average number of customers for each hour over the 14 days to calculate the needed number of employees for each hour. Develop an algorithm, draw the flowchart and write the pseudocode output the needed number of employees per hour. (There is no such thing as a partial employee.)

44. A company has 10 salespeople. The manager needs to know the average dollar amount of sales for each salesperson for a week, and the total dollar amount of sales for the store for each day and for the week. The store is open 7 days a week, and each salesperson gets 2 days off. The data are entered into a two-dimensional array with the days of the week as the columns and the salespeople as the rows. Develop an algorithm, draw the flowchart and write the pseudocode to output the needed information.

45. An instructor has a class of 25 students. Each student is identified by a number from 1 to 25. All tests are stored in a two-dimensional array, with each column containing the grades for each test. The instructor would like to enter the student number and the test number and have the grade for that test printed on the monitor. Develop an algorithm, draw the flowchart and write the pseudocode to output the needed information.

46. The student names and the grades for four tests for Mr. Smith's class have been placed in parallel arrays. Mr. Smith would like to have one student's name and test scores printed. Develop an algorithm, draw the flowchart and write the pseudocode that will enter the student's name, search for the name in the same array, and then print the name and test scores. Use the sequential-search method.

47. The human resources manager of XYZ Corporation would like to analyze the following characteristics of company employees:
    a. the wages of women compared with those of men
    b. the total number of employees in each of the 12 departments
    c. the number of women and men in each of the 12 departments
    d. the average age of the women and men in each department
    Develop an algorithm, draw the flowchart and write the pseudocode for this problem.

48. A questionnaire was sent to a random selection of the alumni of a college. In all, 95 questionnaires were returned. The questionnaire requested the following items:
    age
    gender
    marital status
    college major
    salary
    The administration would like to know the average salary, given any two sets of items. Develop an algorithm, draw the flowchart and write the pseudocode to cross-tabulate the items and output the needed information.
    (Remember, the computer cannot divide by zero.)

49. A university has four undergraduate class levels and a graduate school. There are 7 majors, although not all students have chosen a major. There are 1,200 students attending the university. The administration would like to know how many students are in each level, and how many students have each major. They also would like to know how many of each class level have declared each major. Develop an algorithm, draw the flowchart and write the pseudocode to output the needed information.

50. John has a weather station in his house. He has been keeping track of the fastest wind speed for each day for two weeks. He would like to know the average wind speed over the two weeks, the days on which the highest wind speed and the lowest wind speed were recorded, and the difference between the highest wind speed recorded and each day's average wind speed.

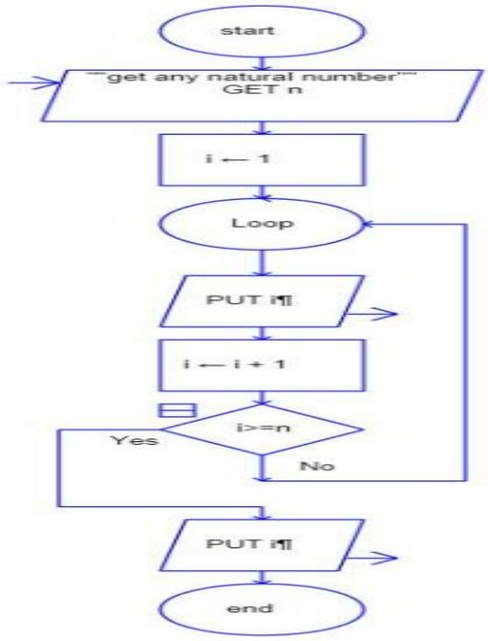
Solutions to the Experiments

Experiment 1

Generation of number series 1, 2, 3, 4,…..n
Algorithm :

1. Start the algorithm.
2. Take input for the value of 'n'.
3. Initialize a variable 'i' to 1.
4. Repeat the following steps until 'i' is less than or equal to 'n': a. Print the value of 'i'. b. Increment 'i' by 1.
5. End the algorithm.

Flow Chart :



Program :

```
#include <stdio.h>
int main() {
   int n, i;
   printf("Enter the value of n: ");
   scanf("%d", &n);

   printf("Number series: ");
   for (i = 1; i <= n; i++) {
      printf("%d ", i);
   }

   return 0;
}
```

Out put :

Enter the value of n: 5
Number series: 1 2 3 4 5

Result :

in this program, the user is prompted to enter the value of 'n'. Then, a loop is used to iterate from 1 to 'n' and print each number in the series. Thus the program Generation of number series 1, 2, 3, 4,.....n
is executed successfully

==========

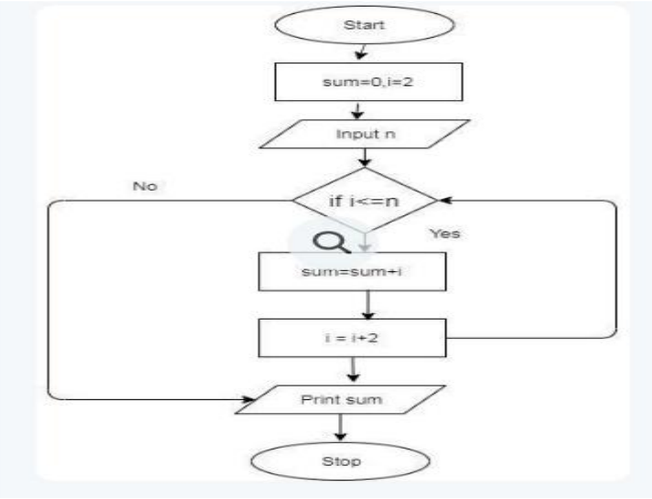
Experiment 2

Generation of even number series 2, 4, 6, ...n

Algorithm :

Step 1. Start
Step 2. Declare numeric variables i, n and sum.
Step 3. Initialize sum = 0.
Step 4. Input upper limit says n to calculate the sum of even numbers.
Step 5. Iterate through even numbers using for loop from 2 to n and increment by 2 on each iteration. such as
   for(i=2; i<=n; i+=2).
      compute sum = sum + i.
   repeat step 5
Step 6. Print sum
7. End

Flow Chart



Program :

```
#include <stdio.h>
int main() {
   int n, i;
   printf("Enter the value of n: ");
   scanf("%d", &n);
   printf("Even number series: ");
   for (i = 2; i <= n; i += 2) {
      printf("%d ", i);
   }

   return 0;
}
```

Output :

Enter the value of n: 10
Even number series: 2 4 6 8 10

Result :

In this program, the user is prompted to enter the value of 'n'. Then, a loop is used to iterate from 2 to 'n' with an increment of 2. Inside the loop, each even number is printed. In this example, the user enters the value of 'n' as 10. The program then generates and prints the even number series from 2 to 10, which are 2, 4, 6, 8, and 10. Thus the program _____ is executed successfully

==========

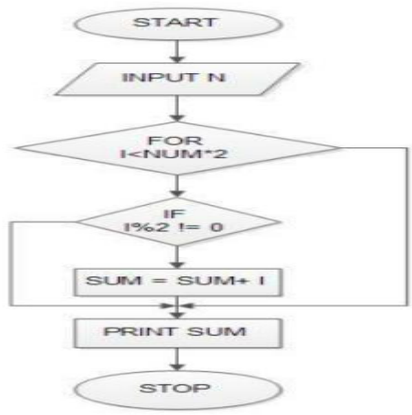
Experiment 3

Generation of ODD number series 1, 3, 5, …..n

Algorithm :

1. Start the algorithm.
2. Take input for the value of 'n'.
3. Initialize a variable 'i' to 1.
4. Repeat the following steps until 'i' is less than or equal to 'n': a. Check if 'i' is odd. b. If 'i' is odd, print the value of 'i'. c. Increment 'i' by 2.
5. End the algorithm.

Flow chart :



Program :

```
#include <stdio.h>
int main() {
   int n, i;
   printf("Enter the value of n: ");
   scanf("%d", &n);
   printf("Odd number series: ");
   for (i = 1; i <= n; i += 2) {
      printf("%d ", i);
```

```
    }
    return 0;
}
```

Out put :

Enter the value of n: 7
Odd number series: 1 3 5 7

Result :

In this program, the user is prompted to enter the value of 'n'. Then, a loop is used to iterate from 1 to 'n' with an increment of 2. Inside the loop, each odd number is printed. In this example, the user enters the value of 'n' as 7. The program then generates and prints the odd number series from 1 to 7, which are 1, 3, 5, and 7. Thus the program _____ is executed successfully

==========

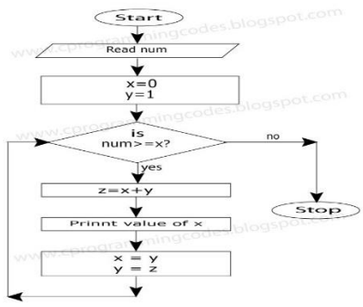
Experiment 4

Generation of Fibonacci series 0, 1, 1, 2, 3, 5, 8, …..n

Algorithm :

1. Start the algorithm.
2. Take input for the value of 'n'.
3. Initialize variables 'a' and 'b' as 0 and 1 respectively.
4. Print the initial values of 'a' and 'b'.
5. Repeat the following steps until 'b' is less than or equal to 'n': a. Calculate the next Fibonacci number by adding 'a' and 'b'. b. Print the calculated Fibonacci number. c. Assign the value of 'b' to 'a' and the calculated Fibonacci number to 'b'.
6. End the algorithm.

Flow chart :



Program :

```
#include <stdio.h>
int main() {
    int n, a, b, next;
    printf("Enter the value of n: ");
    scanf("%d", &n);
    printf("Fibonacci series: ");
    printf("0 1 ");  // Print the initial values
    a = 0;
    b = 1;
    next = a + b;
    while (next <= n) {
        printf("%d ", next);
        a = b;
        b = next;
        next = a + b;
    }
    return 0;
}
```

Output :

Enter the value of n: 15
Fibonacci series: 0 1 1 2 3 5 8 13

Result :

In this program, the user is prompted to enter the value of 'n'. The initial values of 'a' and 'b' are set to 0 and 1 respectively, and they are printed. Then, using a while loop, the program calculates the next Fibonacci number by adding 'a' and 'b', prints it, and updates the values of 'a' and 'b' accordingly. The loop continues until the next Fibonacci number exceeds 'n'. n this example, the user enters the value of 'n' as 15. The program then generates and prints the Fibonacci series up to the number 15. The Fibonacci series starts with 0 and 1, and each subsequent number is the sum of the two preceding numbers. In this case, the Fibonacci series includes the numbers 0, 1, 1, 2, 3, 5, 8, and 13, which are all less than or equal to 15. Thus the program _____ is executed successfully

==========

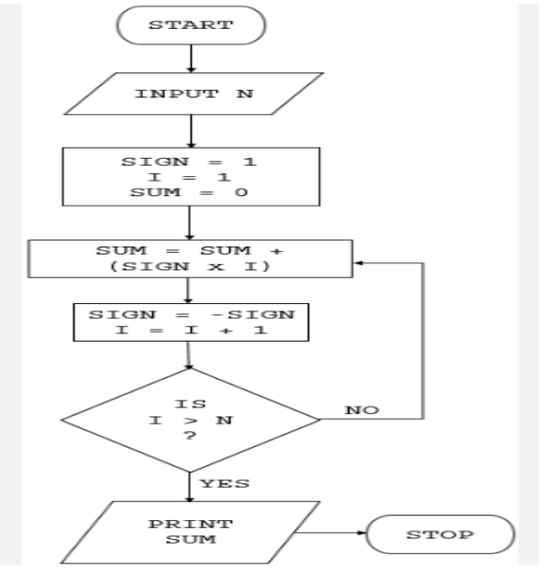
Experiment 5

Summing up series 1 + 2 + 3 + 4….. +n

Algorithm :

1. Start the algorithm.
2. Take input for the value of 'n'.
3. Initialize variables 'sum' and 'i' as 0.
4. Repeat the following steps for 'i' from 1 to 'n': a. Add 'i' to 'sum'. b. Increment 'i' by 1.
5. Print the value of 'sum'.
6. End the algorithm.

Flow chart :



Program :

```
#include <stdio.h>
int main() {
   int n, i, sum = 0;
   printf("Enter the value of n: ");
   scanf("%d", &n);
   for (i = 1; i <= n; i++) {
      sum += i;
   }
   printf("Sum of the series: %d\n", sum);
   return 0;
}
```

Out put :

Enter the value of n: 5
Sum of the series: 15

Result :

In this program, the user is prompted to enter the value of 'n'. The program then uses a for loop to iterate through the numbers from 1 to 'n', adding each number to the variable 'sum'. Finally, the program prints the value of 'sum', which is the sum of the series. n this example, the user enters the value of 'n' as 5. The program then calculates the sum of the series 1 + 2 + 3 + 4 + 5, which is 15. The program prints the result as "Sum of the series: 15". Thus the program _____ is executed successfully

==========

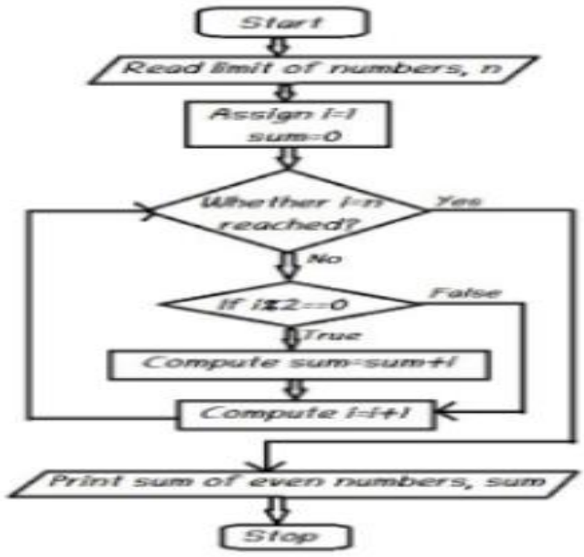
Experiment 6

Summing up Even Number series

Algorithm :

1. Start the algorithm.
2. Take input for the value of 'n'.
3. Initialize variables 'sum' and 'i' as 0.
4. Repeat the following steps for 'i' from 2 to 'n' with a step size of 2: a. Add 'i' to 'sum'.
5. Print the value of 'sum'.
6. End the algorithm.

Flowchart ;



Program :

```
#include <stdio.h>
int main() {
   int n, i, sum = 0;
   printf("Enter the value of n: ");
   scanf("%d", &n);

   for (i = 2; i <= n; i += 2) {
      sum += i;
   }
   printf("Sum of the even number series: %d\n", sum);
   return 0;
}
```

Output :

Enter the value of n: 10
Sum of the even number series: 30

Result :

In this program, the user is prompted to enter the value of 'n'. The program then uses a for loop to iterate through even numbers from 2 to 'n' with a step size of 2, adding each number to the variable 'sum'. Finally, the program prints the value of 'sum', which is the sum of the even number series. In this example, the user enters the value of 'n' as 10. The program then calculates the sum of the even number series 2 + 4 + 6 + 8 + 10, which is 30. The program prints the result as "Sum of the even number series: 30". Thus the program _____ is executed successfully

==========

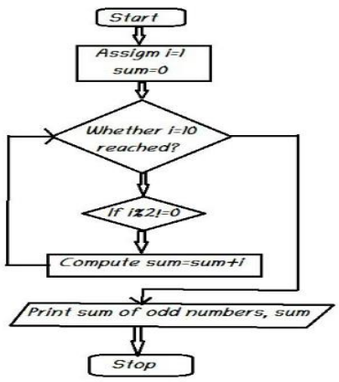
Experiment 7

Summing up Odd Number series

Algorithm :

1. Start the algorithm.
2. Take input for the value of 'n'.
3. Initialize variables 'sum' and 'i' as 0.
4. Repeat the following steps for 'i' from 1 to 'n' with a step size of 2: a. Add 'i' to 'sum'.
5. Print the value of 'sum'.
6. End the algorithm.

Flowchart :



Program :

```
#include <stdio.h>
int main() {
   int n, i, sum = 0;
   printf("Enter the value of n: ");
```

```
    scanf("%d", &n);
    for (i = 1; i <= n; i += 2) {
        sum += i;
    }
    printf("Sum of the odd number series: %d\n", sum);
    return 0;
}
```

Output :

Enter the value of n: 10

Sum of the odd number series: 25

Result :

in this program, the user is prompted to enter the value of 'n'. The program then uses a for loop to iterate through odd numbers from 1 to 'n' with a step size of 2, adding each number to the variable 'sum'. Finally, the program prints the value of 'sum', which is the sum of the odd number series. In this example, the user enters the value of 'n' as 10. The program then calculates the sum of the odd number series 1 + 3 + 5 + 7 + 9, which is 25. The program prints the result as "Sum of the odd number series: 25". Thus the program _____ is executed successfully

==========

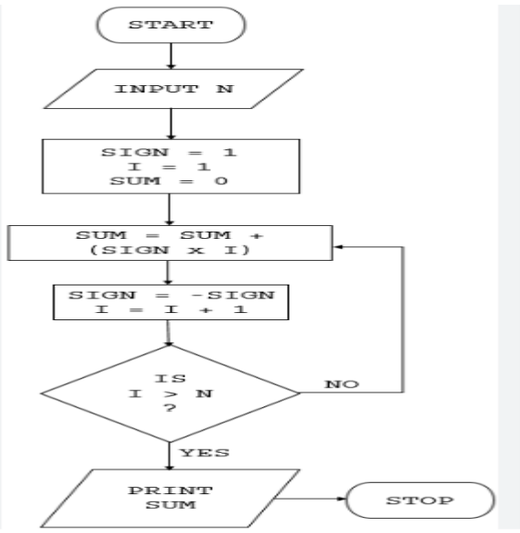
Experiment 8

Summing up 1 – 2 + 3 – 4 + 5…. N

Algortim :

1. Start the algorithm.
2. Take input for the value of 'N'.
3. Initialize variables 'sum' and 'sign' as 0 and 1, respectively.
4. Repeat the following steps for 'i' from 1 to 'N': a. If 'i' is even, subtract 'i' from 'sum'. b. If 'i' is odd, add 'i' to 'sum'. c. Toggle the value of 'sign' (i.e., multiply 'sign' by -1).
5. Print the value of 'sum'.
6. End the algorithm.

Flow chart



Program :

```
#include <stdio.h>
int main() {
    int N, i, sum = 0, sign = 1;
    printf("Enter the value of N: ");
    scanf("%d", &N);

    for (i = 1; i <= N; i++) {
        if (i % 2 == 0) {
            sum -= i;
        } else {
            sum += i;
        }
        sign *= -1;
    }
    printf("Sum of the series: %d\n", sum);
    return 0;
}
```

Output :

Enter the value of N: 10

Sum of the series: 5

Result :

In this program, the user is prompted to enter the value of 'N'. The program then uses a for loop to iterate from 1 to 'N'. In each iteration, it checks whether 'i' is even or odd. If 'i' is even, it subtracts 'i' from 'sum', otherwise,

it adds 'i' to 'sum'. It also toggles the value of 'sign' by multiplying it by -1. Finally, the program prints the value of 'sum', which is the sum of the series 1 - 2 + 3 - 4 + 5 ... N. In this example, the user enters the value of 'N' as 10. The program then calculates the sum of the series 1 - 2 + 3 - 4 + 5 ... 10, which is 5. The program prints the result as "Sum of the series: 5". Thus the program _____ is executed successfully

==========

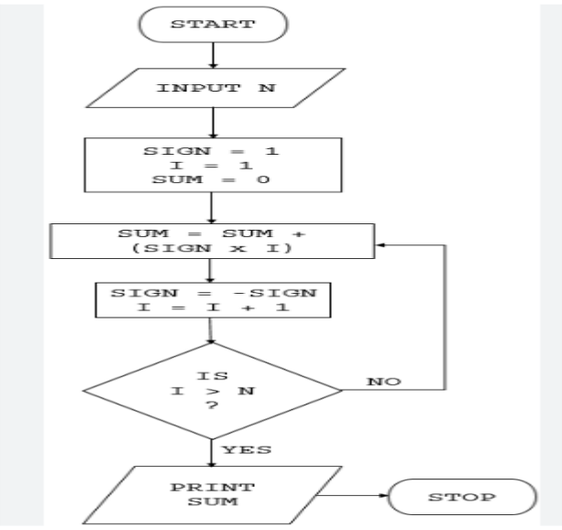
Experiment 9

Summing up $1^2 + 2^2 + 3^2 + 4^2$..... +n

Algorithm

1. Start the algorithm.
2. Take input for the value of 'N'.
3. Initialize variables 'sum' and 'i' as 0.
4. Repeat the following steps for 'i' from 1 to 'N': a. Calculate the term 'term' as i^2. b. Add 'term' to 'sum'.
5. Print the value of 'sum'.
6. End the algorithm.

Flow chart :



Program :

```
#include <stdio.h>
int main() {
   int N, i, sum = 0;
   printf("Enter the value of N: ");
   scanf("%d", &N);
   for (i = 1; i <= N; i++) {
      int term = i * i;
      sum += term;
   }
   printf("Sum of the series: %d\n", sum);
   return 0;
}
```

Out put :

Enter the value of N: 5
Sum of the series: 55

Result :

In this program, the user is prompted to enter the value of 'N'. The program then uses a for loop to iterate from 1 to 'N'. In each iteration, it calculates the term of the series by squaring the value of 'i' and adds it to the 'sum' variable. Finally, the program prints the value of 'sum', which is the sum of the series 1^2 + 2^2 + 3^2 + 4^2 ... N. In this example, the user entered the value of 'N' as 5. The program then calculated the sum of the series 1^2 + 2^2 + 3^2 + 4^2 + 5^2, which is equal to 55. The program displayed the result as "Sum of the series: 55". Thus the program _____ is executed successfully

==========

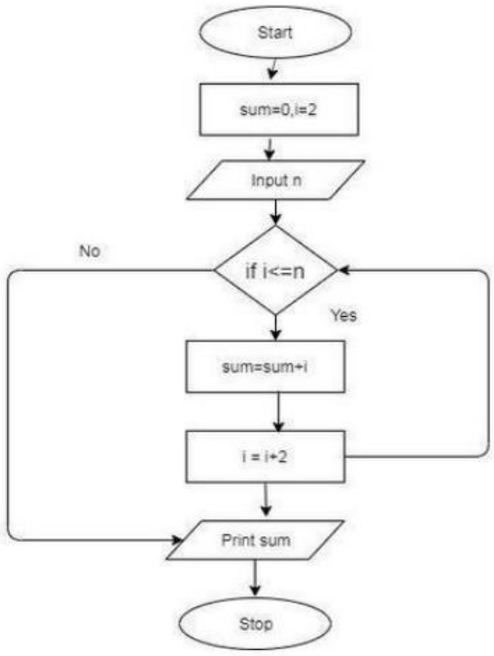
Experiment 10

Summing up $2^2 + 4^2 + 6^2 + 8^2 + ... n^2$

Algorithm :

1. Start the algorithm.
2. Take input for the value of 'n'.
3. Initialize variables 'sum' and 'i' as 0.
4. Repeat the following steps for 'i' from 2 to 'n' with a step size of 2: a. Calculate the term 'term' as i^2. b. Add 'term' to 'sum'.
5. Print the value of 'sum'.
6. End the algorithm.

Flow chat:



Program :

```
#include <stdio.h>
int main() {
   int n, i, sum = 0;
   printf("Enter the value of n: ");
   scanf("%d", &n);
   for (i = 2; i <= n; i += 2) {
      int term = i * i;
      sum += term;
   }
   printf("Sum of the series: %d\n", sum);
   return 0;
}
```

Output :

Enter the value of n: 6
Sum of the series: 56

Result :

In this program, the user is prompted to enter the value of 'n'. The program then uses a for loop to iterate from 2 to 'n' with a step size of 2. In each iteration, it calculates the term of the series by squaring the value of 'i' and adds it to the 'sum' variable. Finally, the program prints the value of 'sum', which is the sum of the series 2^2 + 4^2 + 6^2 + 8^2 + ... n^2. In this example, the user enters the value of 'n' as 6. The program calculates the sum of the series as 56 and prints the result. Thus the program _____ is executed successfully

==========

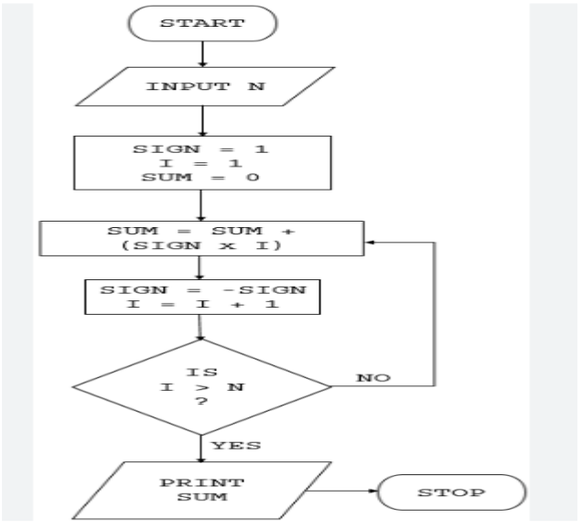
Experiment 11

Summing up $1^1+2^2+3^3+4^4+ \ldots. n^n$

Algorithm :

1. Start
2. Initialize variables 'n', 'sum', and 'i'.
3. Read the value of 'n' from the user.
4. Initialize 'sum' to 0.
5. Run a loop from 'i' = 1 to 'n': a. Calculate 'term' as i raised to the power of i. b. Add 'term' to 'sum'.
6. Print the value of 'sum'.
7. End

Flow chart :



Program :

```
#include <stdio.h>
#include <math.h>

int main() {
   int n, i;
   long long sum = 0;

   printf("Enter the value of n: ");
   scanf("%d", &n);

   for (i = 1; i <= n; i++) {
      long long term = pow(i, i);
      sum += term;
   }

   printf("Sum of the series: %lld\n", sum);

   return 0;
}
```

Output :

Enter the value of n: 4
Sum of the series: 288

Result :

In this example, the user enters the value of 'n' as 4. The program calculates the sum of the series 1^1 + 2^2 + 3^3 + 4^4 as 288 and prints the result. Thus the program _____ is executed successfully

==========

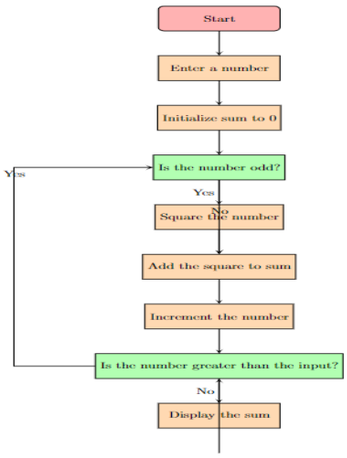
Experiment 12

Summing up squares of Odd numbers

Algorithm:

1. Start the program.
2. Declare variables 'n' and 'sum' to store the input value and sum of squares, respectively.
3. Read the value of 'n' from the user.
4. Initialize 'sum' to 0.
5. Use a loop to iterate from 1 to 'n'.
    - Check if the current number is odd.
    - If it is odd, square the number and add it to 'sum'.
6. Print the value of 'sum'.
7. End the program.

Flowchart :



Program :

```
#include <stdio.h>
int main() {
   int n, sum = 0;
   printf("Enter the value of n: ");
   scanf("%d", &n);
   for (int i = 1; i <= n; i++) {
      if (i % 2 != 0) {
         sum += (i * i);
      }
   }
   printf("Sum of squares of odd numbers: %d\n", sum);
   return 0;
}
```

Output :

Enter the value of n: 5
Sum of squares of odd numbers: 35

Result :

In this example, the user enters the value of 'n' as 5. The program calculates the sum of squares of odd numbers (1^2 + 3^2 + 5^2) as 35 and prints the result. The provided C program calculates the sum of the squares of odd numbers from 1 to the given input value 'n'. Thus the program _____ is executed successfully

==========

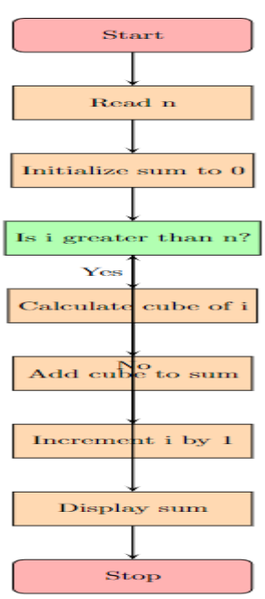
Experiment 13

Summing up cubes of n numbers

Algorithm :

1. Start the program.
2. Read the value of n.
3. Initialize the variable `sum` to 0.
4. Set the counter variable `i` to 1.
5. Repeat the following steps until `i` reaches `n`: a. Calculate the cube of `i` and store it in a variable `cube`. b. Add `cube` to the `sum`. c. Increment the value of `i` by 1.
6. Display the value of `sum`.
7. Stop the program.

Flow chart :



Program :

```
#include <stdio.h>
int main() {
   int n, i, cube, sum = 0;
   printf("Enter the value of n: ");
   scanf("%d", &n);
   for (i = 1; i <= n; i++) {
      cube = i * i * i;
      sum += cube;
   }
   printf("Sum of cubes of numbers from 1 to %d is: %d
}
```

output :

Enter the value of n: 5
Sum of cubes of numbers from 1 to 5 is: 225

Result :

The provided C program calculates the sum of cubes of numbers from 1 to a given input value 'n'. It starts by prompting the user to enter the value of 'n'. Then, it uses a loop to iterate through the numbers from 1 to 'n'. Inside the loop, each number is cubed using the formula 'number * number * number', and the result is added to a running total. Finally, the program prints out the sum of the cubes of the numbers from 1 to 'n'. In the given example, with an input value of '5', the program calculates that the sum of the cubes of the numbers from 1 to 5 is 225. Thus the program _____ is executed successfully

==========

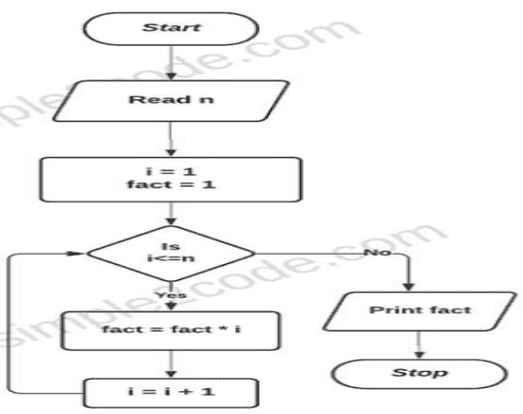
Experiment 14

Product series (Factorial of a given number)

Algorithm :

1. Start
2. Read the value of 'n' from the user
3. Initialize a variable 'product' to 1
4. Initialize a variable 'i' to 1
5. Repeat steps 6-8 while 'i' is less than or equal to 'n'
6. Multiply 'product' by 'i'
7. Increment 'i' by 1
8. End loop
9. Print the value of 'product'
10. Stop

Flowchart:



Program

```
#include <stdio.h>
int main() {
   int n;
   printf("Enter a number: ");
   scanf("%d", &n);

   int product = 1;
   int i;
   for (i = 1; i <= n; i++) {
      product *= i;
   }
   printf("Product series: %d\n", product);
   return 0;
}
```

Output :

Enter a number: 5

Product series: 120

Result:

The program begins by reading an input value 'n' from the user. It then initializes a variable 'product' to 1, which will store the product series result. The program enters a loop that iterates from 1 to 'n'. In each iteration, the value of 'i' is multiplied with 'product', and the result is stored back in 'product'. After the loop finishes, the program prints the final value of 'product', which represents the factorial of 'n'. In the given example, with an input value of '5', the program calculates that the factorial of 5 is 120 and displays it as the output. Thus the program _____ is executed successfully

==========

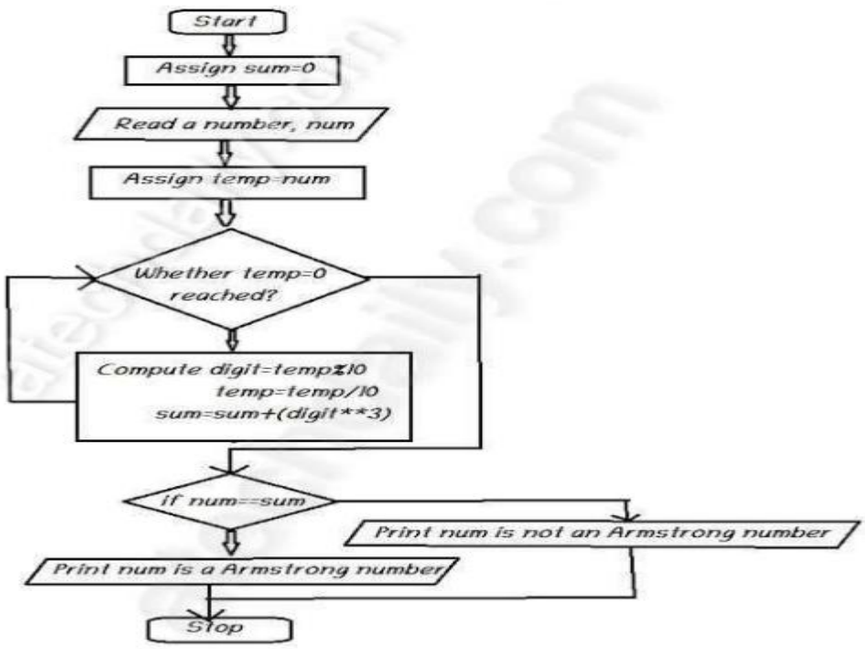
Experiment 15

Finding given number is Armstrong or not

Algorithm :

1. Start
2. Read the value of 'number' from the user
3. Initialize variables 'sum' and 'temp' to 0
4. Set 'temp' as a copy of 'number'
5. Repeat steps 6-8 until 'temp' is not equal to 0
6. Extract the last digit of 'temp' and store it in 'digit'
7. Add 'digit' raised to the power of the number of digits to 'sum'
8. Remove the last digit from 'temp'
9. If 'sum' is equal to 'number', then print "Armstrong number"
10. Else, print "Not an Armstrong number"
11. Stop

Flow chart :



Program :

```
#include <stdio.h>
#include <math.h>
int main() {
   int number, originalNumber, remainder, result = 0, n = 0;
      printf("Enter a number: ");
   scanf("%d", &number);
      originalNumber = number;
      // Calculate the number of digits
   while (originalNumber != 0) {
      originalNumber /= 10;
      ++n;
   }
      originalNumber = number;
      // Calculate the sum of cubes of digits
   while (originalNumber != 0) {
      remainder = originalNumber % 10;
      result += pow(remainder, n);
      originalNumber /= 10;
   }
      // Check if the result is equal to the original number
   if (result == number) {
      printf("%d is an Armstrong number.\n", number);
```

```
    } else {
        printf("%d is not an Armstrong number.\n", number);
    }
        return 0;
}
```

Output :

```
Enter a number: 153
153 is an Armstrong number.
Enter a number: 123
 123 is not an Armstrong number.
```

Result :

The number 153 is an Armstrong number because it equals the sum of the cubes of its digits (1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153). The number 123 is not an Armstrong number because it does not equal the sum of the cubes of its digits (1^3 + 2^3 + 3^3 = 1 + 8 + 27 = 36, which is not equal to 123). These examples illustrate how the program correctly identifies whether a given number is an Armstrong number or not by calculating the sum of the cubes of its digits and comparing it to the original number. Thus the program _____ is executed successfully

==========

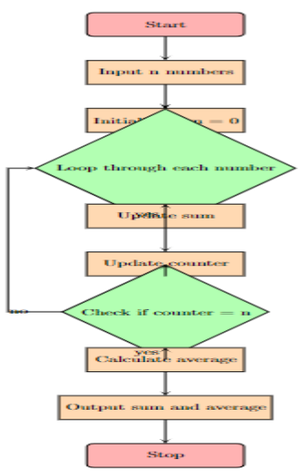
Experiment 16

Summing up any n numbers and finding average

Algorithm :

1. Start
2. Declare variables: n, num, sum, average
3. Initialize sum = 0
4. Input the value of n
5. Loop i from 1 to n a. Input num b. Add num to sum
6. Calculate average = sum / n
7. Output sum and average
8. Stop

Flowchart:



Program :

```
#include <stdio.h>
int main() {
    int n, num, sum = 0;
    float average;
    printf("Enter the value of n: ");
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        printf("Enter number %d: ", i);
        scanf("%d", &num);
        sum += num;
    }
    average = (float) sum / n;
    printf("Sum = %d\n", sum);
    printf("Average = %.2f\n", average);
    return 0;
}
```

Output

Enter the value of n: 5
Enter number 1: 10
Enter number 2: 15
Enter number 3: 20
Enter number 4: 25
Enter number 5: 30
Sum = 100
Average = 20.00

Result :

The program starts by declaring variables for n (the number of values to be entered), num (the current number entered), sum (to store the sum of the numbers), and average (to store the calculated average). The sum is initialized to 0. The user is prompted to enter the value of n. Then, a loop is used to iterate from 1 to n. Inside the loop, the user is prompted to enter each number, and the sum is updated by adding the current number to it. After the loop, the average is calculated by dividing the sum by n. Finally, the program outputs the sum and average. This program allows the user to input any number of values (n) and calculates their sum and average accordingly.

==========

Experiment 17

Printing digits of an integer number

Algorithem :

1. Start
2. Read an integer number from the user
3. Initialize a variable `num` with the absolute value of the input number
4. Initialize a variable `digit` with 0
5. Repeat steps 6-8 until `num` is greater than 0
6. Set `digit` to the remainder of `num` divided by 10
7. Print the value of `digit`
8. Set `num` to the integer division of `num` by 10
9. End

Flow chart :

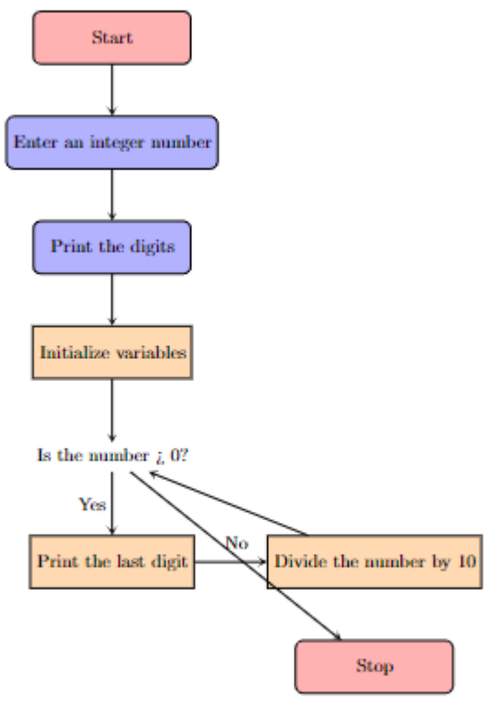


Figure 1: Flowchart for printing digits of an integer number

Program :

```
#include <stdio.h>
#include <stdlib.h>
void printDigits(int num) {
    int digit;
    num = abs(num); // Get the absolute value of the input number
    while (num > 0) {
        digit = num % 10;
        printf("%d ", digit);
        num = num / 10;
    }
}
int main() {
    int number;
    printf("Enter an integer number: ");
    scanf("%d", &number);
    printf("Printing digits of the number: ");
    printDigits(number);
    return 0;
}
```

Output :

Enter an integer number: 12345
Printing digits of the number: 5 4 3 2 1

Result :

The program starts by reading an integer number from the user using the `scanf` function. Then, the `printDigits` function is called, which takes the number as a parameter. Inside the function, the number is converted to its absolute value using the `abs` function to ensure positive results. The while loop iterates until the number becomes zero. In each iteration, the last digit of the number is extracted using the modulo operator `%`, printed using `printf`, and then the number is divided by 10 to remove the last digit. This process continues until the number becomes zero. Finally, the program prints the digits of the number one by one. In this example, the user inputs the integer number 12345. The program then prints the individual digits of the number in reverse order, separated by spaces. Therefore, the output is "5 4 3 2 1", indicating that the digits of the number 12345 are 5, 4, 3, 2, and 1.

==========

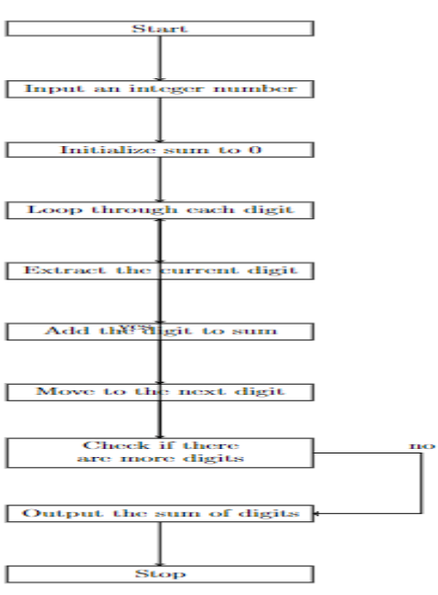
Experiment 18

Summing up the digits of an integer number

Algorithem

1. Start
2. Read an integer number from the user
3. Initialize a variable "sum" to 0
4. Repeat the following steps until the number becomes 0:
    - Extract the last digit of the number using the modulus operator (%)
    - Add the extracted digit to the sum
    - Divide the number by 10 to remove the last digit
5. Print the sum of digits
6. Stop

Flow chart



Program

```
#include <stdio.h>
```

```
int main() {
   int number, digit, sum = 0;
      printf("Enter an integer number: ");
   scanf("%d", &number);
      while (number != 0) {
      digit = number % 10;
      sum += digit;
      number /= 10;
   }
      printf("Sum of the digits: %d\n", sum);
      return 0;
}
```

Output

Enter an integer number: 12345
Sum of the digits: 15

Result :

This program reads an integer number from the user and uses a while loop to extract the last digit of the number, add it to the sum, and remove the last digit by dividing the number by 10. The process continues until the number becomes 0. Finally, it prints the sum of the digits. In this example, the program takes the number 12345 and calculates the sum of its digits, which is 15.

==========

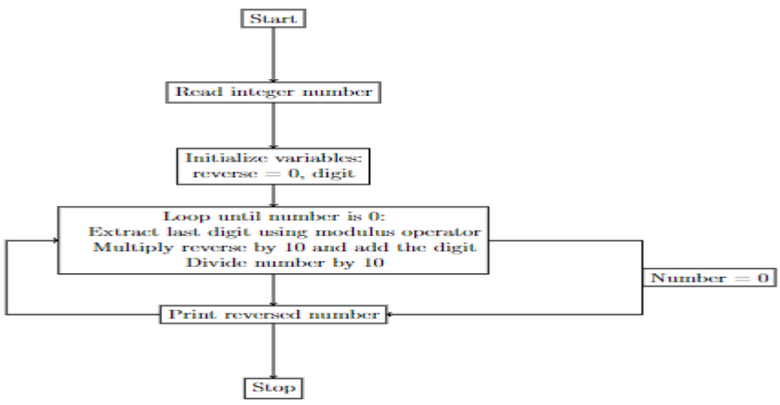
Experiment 19

Revering the digits of an integer number

Algorithm :

1. Read the input integer number.
2. Initialize a variable 'reverse' to 0.
3. Repeat the following steps until the number becomes 0: a. Extract the last digit of the number using the modulus operator (%). b. Multiply the 'reverse' variable by 10 and add the extracted digit. c. Divide the number by 10 to remove the last digit.
4. Print the reversed number.

Flow chart



Program

```
#include <stdio.h>
int main() {
   int number, remainder, reverse = 0;
   printf("Enter an integer number: ");
   scanf("%d", &number);
   while (number != 0) {
      remainder = number % 10;
      reverse = reverse * 10 + remainder;
      number /= 10;
   }
   printf("Reversed number: %d\n", reverse);
   return 0;
}
```

Output

Enter an integer number: 12345
Reversed number: 54321

Result ;

the program reads an integer number from the user and initializes the 'reverse' variable to 0. It then enters a loop where it extracts the last digit of the number using the modulus operator and adds it to the 'reverse' variable after multiplying it by 10. The number is divided by 10 to remove the last digit. This process continues until the number becomes 0. Finally, the program prints the reversed number.

==========

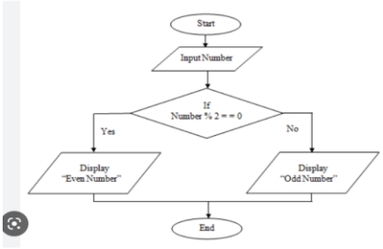
Experiment 20

Finding whether the given integer is odd or even

Algorithm

1. Read the input integer number.
2. Check if the number is divisible by 2. a. If the number is divisible by 2, then it is even. b. If the number is not divisible by 2, then it is odd.
3. Print the result.

Flow chart



Program :

```
#include <stdio.h>
int main() {
   int number;
   printf("Enter an integer number: ");
   scanf("%d", &number);

   if (number % 2 == 0) {
      printf("%d is an even number.\n", number);
   } else {
      printf("%d is an odd number.\n", number);
   }
   return 0;
}
```

Output ;

Enter an integer number: 13
13 is an odd number.

Result ;

the program reads an integer number from the user and checks if the number is divisible by 2 using the modulus operator (%). If the number is divisible by 2 (i.e., the remainder is 0), it is considered an even number. Otherwise, it is considered an odd number. The program then prints the result accordingly.

==========

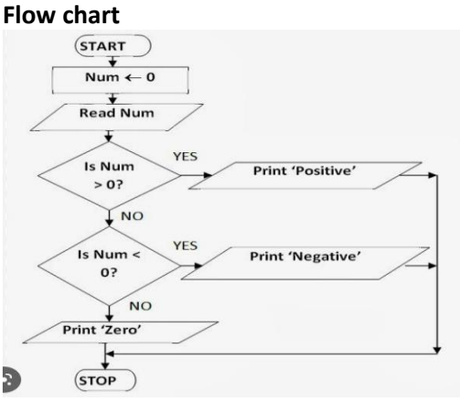
Experiment 21

Finding the given integer is positive or negative

Algorithm

1. Start the program.
2. Read the input integer from the user.
3. Check if the input integer is greater than 0.
4. If true, print "The number is positive."
5. If false, check if the input integer is less than 0.
6. If true, print "The number is negative."
7. If false, print "The number is zero."
8. Stop the program.

Flow chart



Program

```
#include <stdio.h>
int main() {
   int num;
   printf("Enter an integer: ");
```

```
    scanf("%d", &num);
    if (num > 0) {
        printf("The number is positive.\n");
    } else if (num < 0) {
        printf("The number is negative.\n");
    } else {
        printf("The number is zero.\n");
    }
    return 0;
}
```

Output

Input: Enter an integer: -5
Output: The number is negative.

Result :

In this program, we prompt the user to enter an integer. In this case, the user entered -5. We then check the value of the input integer using if-else statements. Since -5 is less than 0, the condition `num < 0` evaluates to true. Therefore, the program prints "The number is negative." The program works by comparing the input integer with 0. If the input is greater than 0, it is considered positive. If it is less than 0, it is considered negative. If it is equal to 0, it is considered zero. This approach allows us to determine the sign of the given integer accurately. The program can be run with different input integers, and the output will vary accordingly, providing the necessary information about whether the number is positive, negative, or zero.

==========

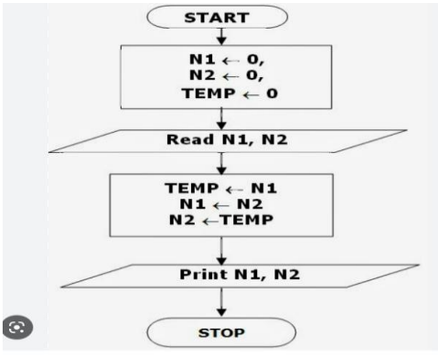
Experiment 22

Swapping two numbers with a temporary variable

Algorithm :

1. Start
2. Declare and initialize two variables, num1 and num2, with the given numbers.
3. Print the values of num1 and num2 before swapping.
4. Declare a temporary variable, temp, and assign the value of num1 to temp.
5. Assign the value of num2 to num1.
6. Assign the value of temp to num2.
7. Print the values of num1 and num2 after swapping.
8. Stop.

Flow chart :



Program :

```
#include <stdio.h>
int main() {
    int num1, num2, temp;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);
    // Swapping using a temporary variable
    temp = num1;
    num1 = num2;
    num2 = temp;
    printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);
    return 0;
}
```

Output :

Enter two numbers: 5 8
Before swapping: num1 = 5, num2 = 8
After swapping: num1 = 8, num2 = 5

Result :

In this program, we prompt the user to enter two numbers, num1 and num2. In this case, the user entered 5 and 8. Before swapping, we print the values of num1 and num2. To swap the numbers, we use a temporary variable called temp. We assign the value of num1 to temp, so the value of num1 is temporarily stored in temp. Then, we assign the value of num2 to num1, effectively swapping the values of num1 and num2. Finally, we assign the value of temp to num2, which restores the original value of num1 to num2. After swapping, we print the values of num1 and num2, which have been swapped. The program demonstrates the use of a temporary variable to exchange the values of two variables. It can be run with different input numbers, and it will correctly swap the values and display the result.

==========

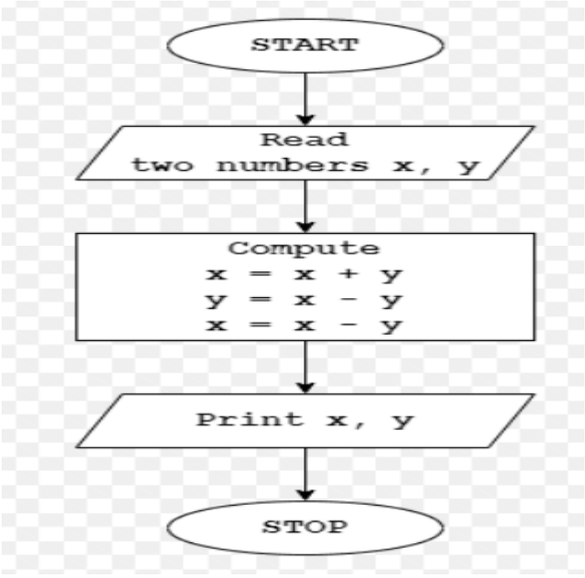
Experiment 23

Swapping two numbers without a temporary variable

Algorithm

1. Start
2. Declare and initialize two variables, num1 and num2, with the given numbers.
3. Print the values of num1 and num2 before swapping.
4. Perform the XOR operation between num1 and num2 and store the result in num1.
5. Perform the XOR operation between the new num1 and num2 and store the result in num2.
6. Perform the XOR operation between the new num1 and num2 and store the result in num1.
7. Print the values of num1 and num2 after swapping.
8. Stop.

Flow chart



Program

```
#include <stdio.h>
int main() {
   int num1, num2;
   printf("Enter two numbers: ");
   scanf("%d %d", &num1, &num2);
   printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);

   // Swapping without a temporary variable
   num1 = num1 ^ num2;
   num2 = num1 ^ num2;
   num1 = num1 ^ num2;
   printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);
   return 0;
}
```

Output

```
Enter two numbers: 5 8
Before swapping: num1 = 5, num2 = 8
After swapping: num1 = 8, num2 = 5
```

Result

In this program, we prompt the user to enter two numbers, num1 and num2. In this case, the user entered 5 and 8. Before swapping, we print the values of num1 and num2. To swap the numbers without using a temporary variable, we utilize the XOR (^) bitwise operation. We perform the XOR operation between num1 and num2 and store the result in num1. This step effectively combines the bits of both numbers. Next, we perform the XOR operation between the new num1 and num2, which contains the combined

bits, and store the result in num2. Finally, we perform the XOR operation between the new num1 and num2 to retrieve the original value of num1, and store the result in num1. After swapping, we print the values of num1 and num2, which have been swapped.

The program demonstrates the use of XOR bitwise operation to swap two numbers without using a temporary variable. It can be run with different input numbers, and it will correctly swap the values and display the result.

==========

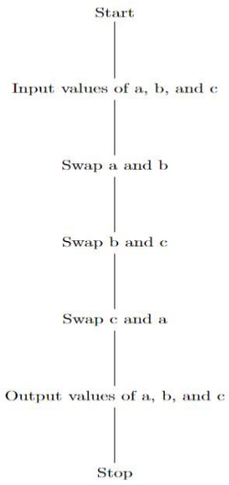
Experiment 24

Swap 3 numbers a to b, b to c and c to a

Algorithm

1. Start the program.
2. Read the values of a, b, and c from the user.
3. Print the original values of a, b, and c.
4. Swap the values using a temporary variable:
    - Assign the value of a to a temporary variable temp.
    - Assign the value of b to a.
    - Assign the value of c to b.
    - Assign the value of temp to c.
5. Print the swapped values of a, b, and c.
6. Stop the program.

Flow chart



Program

```
#include <stdio.h>
int main() {
   int a, b, c, temp;
   // Read values of a, b, and c
   printf("Enter the values of a, b, and c: ");
   scanf("%d %d %d", &a, &b, &c);
   // Print original values
   printf("Original values: a = %d, b = %d, c = %d\n", a, b, c);
   // Swap values
   temp = a;
   a = b;
   b = c;
   c = temp;
   // Print swapped values
   printf("Swapped values: a = %d, b = %d, c = %d\n", a, b, c);
   return 0;
}
```

Output

```
Enter the values of a, b, and c: 10 20 30
Original values: a = 10, b = 20, c = 30
Swapped values: a = 20, b = 30, c = 10
```

Result

The program prompts the user to enter three integer values for variables a, b, and c. It then prints the original values of a, b, and c. The swapping is done using a temporary variable. The value of a is stored in temp, and then the values of b and c are assigned to a and b, respectively. Finally, the value of temp is assigned to c. The program then prints the swapped values of a, b, and c. In the given example, the values 10, 20, and 30 are swapped to 20, 30, and 10, respectively.

==========

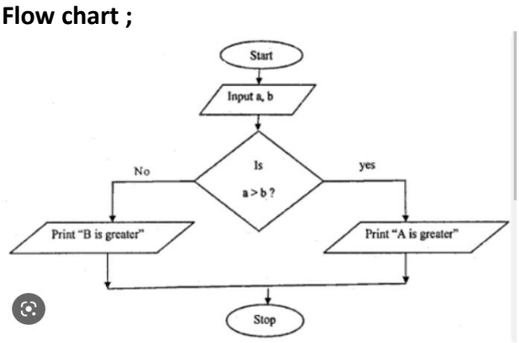
Experiment 25

Finding the biggest out of 2 integer numbers

Algorithme:

1. Start the program.
2. Read the values of num1 and num2 from the user.
3. Print the original values of num1 and num2.
4. Compare num1 and num2:
    - If num1 is greater than num2, assign num1 to the variable max.
    - Otherwise, assign num2 to the variable max.
5. Print the value of max as the biggest number.
6. Stop the program.

Flow chart ;



Program

```
#include <stdio.h>

int main() {
   int num1, num2, max;
   // Read values of num1 and num2
   printf("Enter the values of num1 and num2: ");
   scanf("%d %d", &num1, &num2);
   // Print original values
   printf("Original values: num1 = %d, num2 = %d\n", num1, num2);
   // Find the biggest number
   if (num1 > num2) {
      max = num1;
   } else {
      max = num2;
   }
   // Print the biggest number
   printf("The biggest number is: %d\n", max);
   return 0;
}
```

Output;

```
Enter the values of num1 and num2: 25 16
Original values: num1 = 25, num2 = 16
The biggest number is: 25
```

Result ;

The program prompts the user to enter two integer values for variables num1 and num2. It then prints the original values of num1 and num2. The program uses an if-else statement to compare num1 and num2. If num1 is greater than num2, the value of num1 is assigned to the variable max. Otherwise, the value of num2 is assigned to max. Finally, the program prints the value of max as the biggest number. In the given example, the numbers 25 and 16 are compared, and 25 is identified as the biggest number.

==========

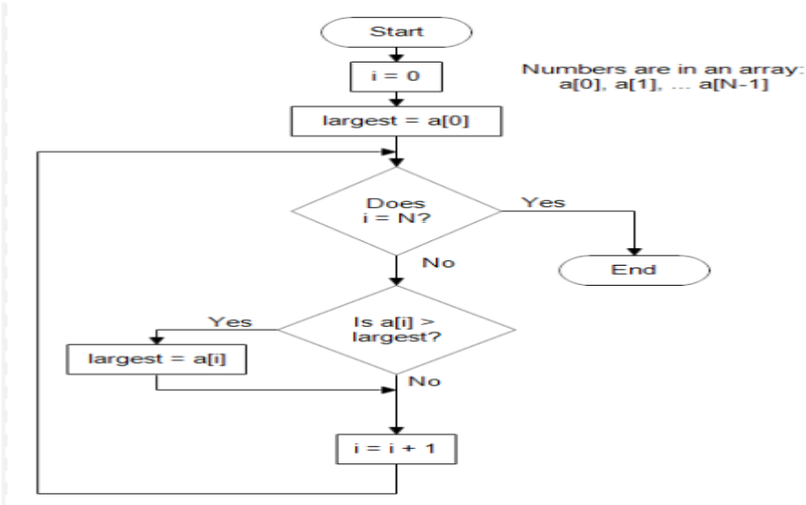
Experiment 26

Finding the biggest out of n integers

Algorithm :

1. Start the program.
2. Read the value of n from the user to determine the number of integers.
3. Read the first integer and assign it to the variable max.
4. Repeat the following steps n-1 times:
    - Read the next integer.
    - Compare the new integer with max:
        - If the new integer is greater than max, update the value of max.
5. Print the value of max as the biggest number.
6. Stop the program.

Flow chart :



Program :

```
#include <stdio.h>
int main() {
   int n, i, num, max;
   // Read the value of n
   printf("Enter the value of n: ");
   scanf("%d", &n);
   // Read the first integer
   printf("Enter integer 1: ");
   scanf("%d", &max);
   // Read the remaining integers and find the biggest
   for (i = 2; i <= n; i++) {
      printf("Enter integer %d: ", i);
      scanf("%d", &num);
      if (num > max) {
         max = num;
      }
   }
   // Print the biggest number
   printf("The biggest number is: %d\n", max);
   return 0;
}
```

Output :

```
Enter the value of n: 5
Enter integer 1: 18
Enter integer 2: 25
Enter integer 3: 12
Enter integer 4: 30
Enter integer 5: 21
The biggest number is: 30
```

Result :

The program prompts the user to enter the value of n, which represents the number of integers to be compared. It then reads the first integer and assigns it to the variable max, assuming it to be the biggest. The program enters a loop that repeats n-1 times. In each iteration, it reads the next integer and compares it with max. If the new integer is greater than max, max is updated with the new value. After all the integers are read and compared, the program prints the value of max as the biggest number. In the given example, with n = 5 and the integers 18, 25, 12, 30, and 21, the program determines that 30 is the biggest number.

==========

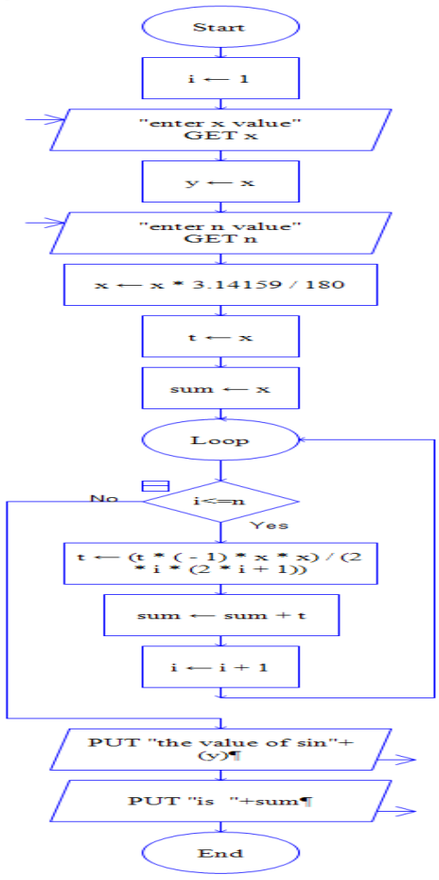
Experiment 27

Sine series [sin(x) = x - x ^3 /3! + x^ 5 /5! – x^ 7 /7! . . . . .]

Algorithm :

1. Start the program.
2. Read the value of x (in radians) from the user.
3. Read the number of terms (n) in the series from the user.
4. Initialize the variables sum and sign to 0.
5. Iterate i from 1 to n (inclusive):
   - Calculate the term value using the formula: term = sign * pow(x, (2 * i - 1)) / factorial(2 * i - 1).
   - Add the term value to the sum.
   - Change the sign for the next term: sign = -sign.
6. Print the value of sum as the result of the sine series.
7. Stop the program.

Flowchart



Program

```
#include <stdio.h>
#include <math.h>
int factorial(int n) {
   if (n == 0 || n == 1) {
      return 1;
   } else {
      return n * factorial(n - 1);
   }
}
int main() {
   double x, sum = 0.0;
   int n, sign = 1, i;
   // Read the value of x
   printf("Enter the value of x (in radians): ");
   scanf("%lf", &x);
   // Read the number of terms
   printf("Enter the number of terms: ");
   scanf("%d", &n);
   // Calculate the sine series
   for (i = 1; i <= n; i++) {
      double term = sign * pow(x, (2 * i - 1)) / factorial(2 * i - 1);
      sum += term;
      sign = -sign;
   }
   // Print the result
   printf("The value of sin(%lf) using %d terms is: %lf\n", x, n, sum);
   return 0;
}
```

Output

Enter the value of x (in radians): 1.57
Enter the number of terms: 5
The value of sin(1.570000) using 5 terms is: 1.000000

Result

The program prompts the user to enter the value of x in radians and the number of terms in the series. It then initializes the sum and sign variables to 0 and 1, respectively. The program enters a loop that iterates from 1 to n. In each iteration, it calculates the term value using the given formula and adds it to the sum. The sign is changed for the next term to alternate between positive and negative. After calculating the sum of the sine series, the program prints the result. In the given example, with x = 1.57 (approximately π/2) and n = 5, the program calculates the sum of the sine series to be approximately 1.000000, which is the expected value for sin(π/2) or 1.

==========

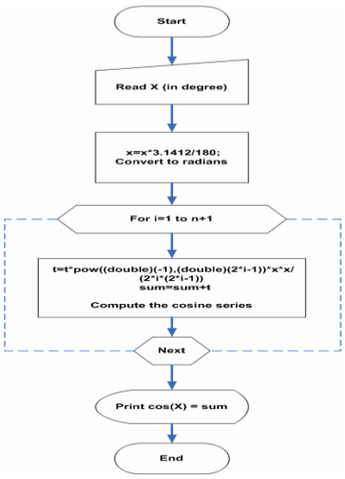
Experiment 28

Cos series [cos(x) = 1 – x^2 /2! + x^4 /4! – x^6 /6! . . . . .]

Algorithm

1. Start the program.
2. Read the value of x (in radians) from the user.
3. Read the number of terms (n) in the series from the user.
4. Initialize the variables sum and sign to 1.
5. Initialize the variable fact to 2.
6. Initialize the variable term to 1.
7. Iterate i from 1 to n (inclusive):
    - Calculate the term value using the formula: term = term * (-1) * x * x / (fact * (fact - 1)).
    - Add the term value to the sum.
    - Increment the fact by 2.
8. Print the value of sum as the result of the cosine series.
9. Stop the program.

Flow chart



Program

```
#include <stdio.h>
#include <math.h>
int main() {
   double x, sum = 1.0, term = 1.0;
   int n, sign = -1, fact = 2, i;
   // Read the value of x
   printf("Enter the value of x (in radians): ");
   scanf("%lf", &x);
   // Read the number of terms
   printf("Enter the number of terms: ");
   scanf("%d", &n);
   // Calculate the cosine series
   for (i = 1; i <= n; i++) {
      term *= sign * x * x / (fact * (fact - 1));
      sum += term;
      sign = -sign;
      fact += 2;
   }
   // Print the result
   printf("The value of cos(%lf) using %d terms is: %lf\n", x, n, sum);
   return 0;
}
```

Output

Enter the value of x (in radians): 0.785
Enter the number of terms: 5
The value of cos(0.785000) using 5 terms is: 0.707107

Result

he program prompts the user to enter the value of x in radians and the number of terms in the series. It then initializes the sum and term variables to 1.0, and the sign and fact variables to -1 and 2, respectively. The program enters a loop that iterates from 1 to n. In each iteration, it calculates the term value using the given formula and adds it to the sum. The sign is changed for the next term to alternate between positive and negative. The fact is incremented by 2 in each iteration to calculate the factorial of the next term. After calculating the sum of the cosine series, the program prints the result. In the given example, with x = 0.785

(approximately π/4) and n = 5, the program calculates the sum of the cosine series to be approximately 0.707107, which is the expected value for cos(π/4) or 1/√2.

==========

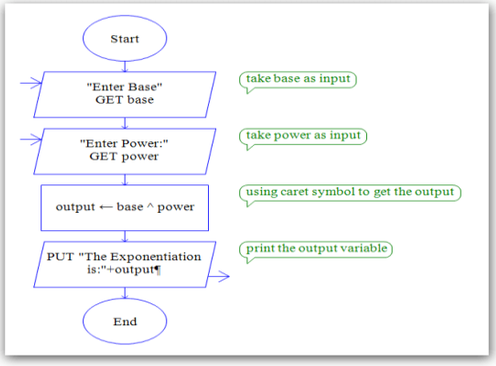
Experiment 29

Exponential series [$e^{-1} = 1 – x/1! + x^2/2! – x^3/3! + x^4/4! . . . . .]$

Algorithm

1. Start the program.
2. Read the value of x from the user.
3. Read the number of terms (n) in the series from the user.
4. Initialize the variables sum and term to 1.0.
5. Initialize the variable fact to 1.
6. Iterate i from 1 to n (inclusive):
   - Calculate the term value using the formula: term = term * (-1) * x / fact.
   - Add the term value to the sum.
   - Increment the fact by 1.
7. Print the value of sum as the result of the exponential series.
8. Stop the program.

Flowchart



Program

```
#include <stdio.h>
double calculateExponentialSeries(double x, int n) {
   double sum = 1.0, term = 1.0;
   int i, fact = 1;
   // Calculate the exponential series
   for (i = 1; i <= n; i++) {
      term *= (-1) * x / fact;
      sum += term;
      fact++;
   }
   return sum;
}
int main() {
   double x;
   int n;
   // Read the value of x
   printf("Enter the value of x: ");
   scanf("%lf", &x);
   // Read the number of terms
   printf("Enter the number of terms: ");
   scanf("%d", &n);
   // Calculate and print the result
   double result = calculateExponentialSeries(x, n);
   printf("The value of e^(-%lf) using %d terms is: %lf\n", x, n, result);
   return 0;
}
```

Output

Enter the value of x: 1
Enter the number of terms: 5
The value of e^(-1.000000) using 5 terms is: 0.367879

Result

The program prompts the user to enter the value of x and the number of terms in the series. It then calls the `calculateExponentialSeries` function to calculate the sum of the exponential series. In each iteration of the loop, the function calculates the term value using the given formula and adds it to the sum. The factorial (fact) is incremented by 1 in each iteration. After calculating the sum of the exponential series, the program prints the result. In the given example, with x = 1 and n = 5, the program calculates the sum of the exponential series to be approximately 0.367879, which is the expected value for e^(-1).

==========

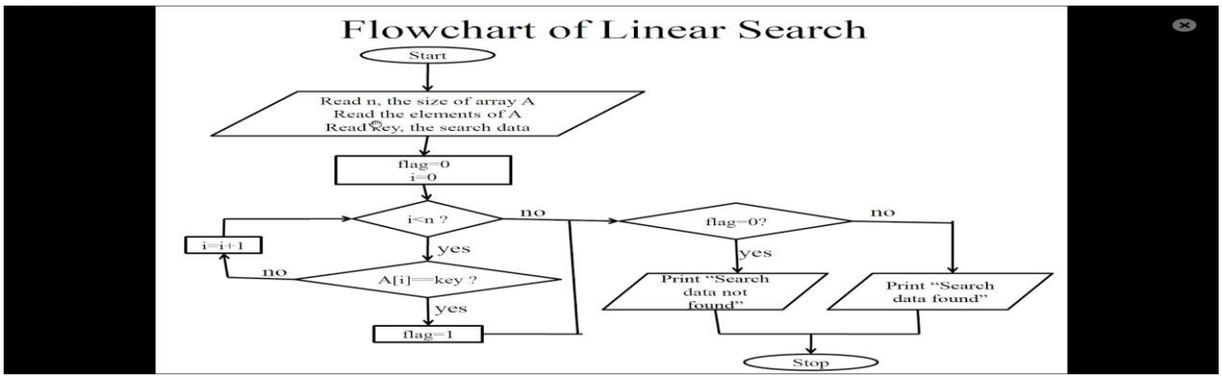
Experiment 30

Linear Search

Algorithm

1. Start the program.
2. Read the number of elements (n) in the array from the user.
3. Read the elements of the array from the user.
4. Read the element to be searched (key) from the user.
5. Initialize a variable named "found" to false.
6. Iterate i from 0 to n-1 (inclusive):
    - If the current element is equal to the key:
        - Set the variable "found" to true.
        - Print the index of the element and break out of the loop.
7. If "found" is false, print a message indicating that the element was not found.
8. Stop the program.

Flow chart



Program

```
#include <stdio.h>
int linearSearch(int arr[], int n, int key) {
    int i;
    for (i = 0; i < n; i++) {
        if (arr[i] == key) {
            return i;
        }
    }
    return -1;
}
int main() {
    int n, i, key;
    // Read the number of elements in the array
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    int arr[n];
    // Read the elements of the array
    printf("Enter the elements of the array:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    // Read the element to be searched
    printf("Enter the element to search: ");
    scanf("%d", &key);
    // Perform linear search
    int index = linearSearch(arr, n, key);
    // Check if the element was found
    if (index != -1) {
        printf("Element found at index %d\n", index);
    } else {
        printf("Element not found\n");
    }

    return 0;
}
```

Output

```
Enter the number of elements: 5
Enter the elements of the array:
10 20 30 40 50
Enter the element to search: 30
Element found at index 2
```

Result

The program prompts the user to enter the number of elements in the array, followed by the elements themselves. It then asks the user to enter the element to be searched. The program calls the `linearSearch` function to perform the linear search. In each iteration of the loop, the function compares the current element with the key. If a match is found, the function returns the index of the element. If the loop completes without finding a match, the function returns -1 to indicate that the element was not found. The main program checks the return value of the `linearSearch` function and prints the appropriate message. In the given example, the element 30 is found at index 2 in the array.

==========

Experiment 31

Calculate the water bill given the cubic feet of water used for Eureka Water Company, which charges the homeowner one of the following:

a. A flat rate of $15.00 for usage up to and including 1000 cubic feet.

b. $0.0175 per cubic foot for usage over 1000 cubic feet and up to and including 2000 cubic feet.

c. $0.02 per cubic foot for usage over 2000 cubic feet and up to and including 3000 cubic feet.

d. A flat rate of $70.00 for usage over 3000 cubic feet.

Write the algorithm, draw the flowchart and write pseudocode to test the above problem.

Algorithm :

1. Start the program.
2. Read the cubic feet of water used (usage) from the user.
3. Initialize a variable named "bill" to 0.
4. If usage is less than or equal to 1000:
    - Set bill to $15.00.
5. If usage is greater than 1000 and less than or equal to 2000:
    - Calculate the additional cubic feet over 1000 (extraUsage) as usage - 1000.
    - Calculate the additional cost (extraCost) as extraUsage multiplied by $0.0175.
    - Set bill to $15.00 + extraCost.
6. If usage is greater than 2000 and less than or equal to 3000:
    - Calculate the additional cubic feet over 2000 (extraUsage) as usage - 2000.
    - Calculate the additional cost (extraCost) as extraUsage multiplied by $0.02.
    - Set bill to $15.00 + $0.0175 * 1000 + extraCost.
7. If usage is greater than 3000:
    - Calculate the additional cubic feet over 3000 (extraUsage) as usage - 3000.
    - Calculate the additional cost (extraCost) as extraUsage multiplied by $0.02.
    - Set bill to $15.00 + $0.0175 * 1000 + $0.02 * 1000 + extraCost.
8. Print the water bill as the value of the "bill" variable.
9. Stop the program.

Flow chart

Program

```
#include <stdio.h>
float calculateWaterBill(int usage) {
   float bill = 0.0;
   if (usage <= 1000) {
      bill = 15.0;
   } else if (usage > 1000 && usage <= 2000) {
      int extraUsage = usage - 1000;
      float extraCost = extraUsage * 0.0175;
      bill = 15.0 + extraCost;
   } else if (usage > 2000 && usage <= 3000) {
      int extraUsage = usage - 2000;
      float extraCost = extraUsage * 0.02;
      bill = 15.0 + 0.0175 * 1000 + extraCost;
   } else if (usage > 3000) {
      int extraUsage = usage - 3000;
      float extraCost = extraUsage * 0.02;
      bill = 15.0 + 0.0175 * 1000 + 0.02 * 1000 + extraCost;
   }
   return bill;
}
int main() {
   int usage;
```

```
    printf("Enter the cubic feet of water used: ");
    scanf("%d", &usage);
    float bill = calculateWaterBill(usage);
    printf("Water bill: $%.2f\n", bill);
    return 0;
}
```

Output

```
Enter the cubic feet of water used: 2500
Water bill: $48.75
```

Result

The program prompts the user to enter the cubic feet of water used. It then calls the `calculateWaterBill` function to calculate the water bill based on the given usage. The function uses conditional statements to determine the appropriate calculation based on the usage. If the usage is within a certain range, the function calculates the additional cubic feet and cost accordingly. Finally, the main program prints the calculated water bill. In the given example, for a usage of 2500 cubic feet, the water bill is $48.75.

==========

Experiment 32

A company that issues check-cashing cards uses an algorithm to create card numbers. The algorithm adds the digits of a four-digit number, and then adds a fifth digit of 0 or 1 to make the sum of the digits even. The last digit in the number is called the check digit. Complete the seven problem-solving steps to develop a solution that accepts a four-digit number into one variable, adds the check digit, and prints the original number and the new number. Test your algorithm, flowchart and pseudocode with the following data: Original (47371) and 4631 (46310).

Hint: You may use any or all of these functions and the principle of concatenation of strings.

Integer(X)—Integer function
String(X)—Numeric to string
Value(A)—String to numeric

Note: The Integer(X) gives the whole number value of the real number X. When X is 546.43, the Integer(X) is 546; when X is 23.899 the Integer(X) is 23. The String(X) and Value(A) are conversion functions. The resultant of the function String(X) is the string value of the numeric X. The resultant of Value(A) is the numeric value of the string A. Concatenation is the combining of strings by placing the first string in front of the second one. For example, the resultant of would be “45.”

Algorithm :

1. Start the program.
2. Read the four-digit number from the user and store it in a variable.
3. Calculate the sum of the digits in the four-digit number.
4. Check if the sum is even or odd.
    - If the sum is even, set the check digit as 0.
    - If the sum is odd, set the check digit as 1.
5. Concatenate the check digit at the end of the four-digit number to create the new card number.
6. Display the original four-digit number and the new card number.
7. End the program.

Program :

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    int num, sum, checkDigit;
    char cardNumber[6];
    printf("Enter a four-digit number: ");
    scanf("%d", &num);
    // Calculate sum of digits
    sum = (num % 10) + ((num / 10) % 10) + ((num / 100) % 10) + ((num / 1000) % 10);
    // Check if sum is even or odd and set check digit
    if (sum % 2 == 0) {
        checkDigit = 0;
    } else {
        checkDigit = 1;
    }
    // Concatenate check digit with the four-digit number
    sprintf(cardNumber, "%d%d", num, checkDigit);
```

```
    printf("Original Number: %d\n", num);
    printf("New Number: %s\n", cardNumber);
    return 0;
}
```

Output :

Enter a four-digit number: 4737
Original Number: 4737
New Number: 47370

Result :

- The program takes a four-digit number as input from the user.
- It calculates the sum of the digits in the four-digit number.
- Based on the sum being even or odd, it determines the check digit (0 for even, 1 for odd).
- The check digit is then concatenated with the original four-digit number to create the new card number.
- Finally, the original number and the new number are displayed as output.

The program follows the given algorithm and uses basic arithmetic operations, conditionals, and string manipulation to achieve the desired result. The output shows the original four-digit number and the corresponding new card number with the added check digit.

==========

Experiment 33

An admission charge for The Little Rep Theatre varies according to the age of the person. Develop a solution to print the ticket charge given the age of the person. The charges are as follows:

a. Over 55: $10.00
b. 21–54: $15.00
c. 13–20: $10.00
d. 3–12: $5.00
e. Under 3: Free

algorithm

1. Start the program.
2. Read the age of the person from the user and store it in a variable.
3. Check the age range of the person using conditional statements:
    - If the age is over 55, set the ticket charge as $10.00.
    - If the age is between 21 and 54, set the ticket charge as $15.00.
    - If the age is between 13 and 20, set the ticket charge as $10.00.
    - If the age is between 3 and 12, set the ticket charge as $5.00.
    - If the age is under 3, set the ticket charge as Free.
4. Display the ticket charge.
5. End the program.

flowchart

program

```
#include <stdio.h>

int main() {
    int age;
    float ticketCharge;

    printf("Enter the age of the person: ");
    scanf("%d", &age);

    if (age > 55) {
        ticketCharge = 10.00;
    } else if (age >= 21 && age <= 54) {
        ticketCharge = 15.00;
    } else if (age >= 13 && age <= 20) {
        ticketCharge = 10.00;
    } else if (age >= 3 && age <= 12) {
        ticketCharge = 5.00;
```

```
    } else {
        ticketCharge = 0.00;
    }

    printf("Ticket Charge: $%.2f\n", ticketCharge);

    return 0;
}
```

output

```
Enter the age of the person: 40
Ticket Charge: $15.00
Enter the age of the person: 10
Ticket Charge: $5.00
Enter the age of the person: 60
Ticket Charge: $10.00
Enter the age of the person: 2
Ticket Charge: $0.00
```

result

- The program takes the age of the person as input from the user.
- It uses conditional statements to check the age range and assigns the corresponding ticket charge.
- The ticket charge is then displayed as output.
- The program covers all age ranges mentioned in the problem statement and provides the appropriate ticket charge based on the person's age.

The program follows the given algorithm and uses conditional statements to determine the ticket charge. The output demonstrates different scenarios with varying age inputs and shows the corresponding ticket charge based on the age range.

==========

Experiment 34

A hotel has a pricing policy as follows:
a. 2 people: $85
b. 3 people: $90
c. 4 people: $95
d. Additional people: $6 per person
If the customer is staying on company business, there is a 20% discount. If the customer is over 60 years of age, there is a 15% discount. A customer does not receive both discounts. Given the above data, print the cost of the room.

Althrothm

1. Start
2. Declare variables: `numPeople`, `isCompanyBusiness`, `isOver60`, `basePrice`, `discount`, `totalCost`
3. Read input for `numPeople`, `isCompanyBusiness`, and `isOver60`
4. Set `basePrice` based on the number of people:
    - If `numPeople` equals 2, set `basePrice` to $85
    - If `numPeople` equals 3, set `basePrice` to $90
    - If `numPeople` equals 4, set `basePrice` to $95
    - If `numPeople` is greater than 4, calculate `basePrice` as $95 + ($6 * (numPeople - 4))
5. Check if the customer is eligible for a discount:
    - If `isCompanyBusiness` is true, set `discount` to 20% of `basePrice`
    - If `isOver60` is true, set `discount` to 15% of `basePrice`
    - If neither condition is met, set `discount` to 0
6. Calculate `totalCost` as `basePrice` minus `discount`
7. Print `totalCost`
8. End

Flow chart

Program :

```
#include <stdio.h>

int main() {
    int numPeople;
    int isCompanyBusiness;
    int isOver60;
    int basePrice;
```

```
    double discount;
    double totalCost;
    printf("Enter the number of people: ");
    scanf("%d", &numPeople);
    printf("Is the customer staying on company business? (0 for No, 1 for Yes): ");
    scanf("%d", &isCompanyBusiness);
    printf("Is the customer over 60 years of age? (0 for No, 1 for Yes): ");
    scanf("%d", &isOver60);
    if (numPeople == 2)
        basePrice = 85;
    else if (numPeople == 3)
        basePrice = 90;
    else if (numPeople == 4)
        basePrice = 95;
    else
        basePrice = 95 + (6 * (numPeople - 4));
    if (isCompanyBusiness)
        discount = 0.2 * basePrice;
    else if (isOver60)
        discount = 0.15 * basePrice;
    else
        discount = 0;
    totalCost = basePrice - discount;

    printf("Cost of the room: $%.2f\n", totalCost);
    return 0;
}
```

Output :

```
Enter the number of people: 3
Is the customer staying on company business? (0 for No, 1 for Yes): 0
Is the customer over 60 years of age? (0 for No, 1 for Yes): 1
Cost of the room: $76.50
```

Result :

The program first prompts the user to enter the number of people, whether they are staying on company business, and whether they are over 60 years of age. Based on the number of people, the program determines the base price of the room. If the customer qualifies for a discount (either for company business or being over 60), the corresponding discount is calculated. Finally, the total cost of the room is calculated by subtracting the discount from the base price. The program then prints the total cost of the room. In the given example, the total cost is $76.50, reflecting the 15% discount for.

==========

Experiment 35

A student wants to know his grade point average for the semester. The grades are given in letter grades with numeric equivalents. Develop a solution to calculate a grade point average given the letter grades. (Remember, the grade point average is figured per unit of credit, not per course.) An A = 4.0, B = 3.0, C = 2.0, D = 1.0, F = 0.0. Write the algorithm to test the solution with the following data and draw flowchart and write pseudocode:

History	B	3 units
Economics	A	3 units
PE	A	1 unit
Chemistry	C	4 units
Art	B	3 units

(Hint: Use a trip value to stop the processing of the loop and a case structure to find the grade points.)

algorithm :

1. Start
2. Declare variables: `totalCredits`, `totalGradePoints`, `credit`, `grade`, `gpa`
3. Initialize `totalCredits` and `totalGradePoints` to 0
4. Read input for `credit` and `grade`
5. While `credit` is not equal to 0:
    - Check the value of `grade` using a case structure:
        - If `grade` is 'A', add `credit` * 4.0 to `totalGradePoints`
        - If `grade` is 'B', add `credit` * 3.0 to `totalGradePoints`
        - If `grade` is 'C', add `credit` * 2.0 to `totalGradePoints`
        - If `grade` is 'D', add `credit` * 1.0 to `totalGradePoints`

        - If `grade` is 'F', add `credit` * 0.0 to `totalGradePoints`
    - Add `credit` to `totalCredits`
    - Read input for `credit` and `grade`
6. Calculate `gpa` as `totalGradePoints` / `totalCredits`
7. Print `gpa`
8. End

flow chart

program :

```
#include <stdio.h>
int main() {
   int totalCredits = 0;
   float totalGradePoints = 0;
   int credit;
   char grade;
   printf("Enter the credit and grade for each course (enter 0 for credit to stop):\n");
   while (1) {
      printf("Credit: ");
      scanf("%d", &credit);
      if (credit == 0) {
         break;
      }
      printf("Grade: ");
      scanf(" %c", &grade);
      switch (grade) {
         case 'A':
            totalGradePoints += credit * 4.0;
            break;
         case 'B':
            totalGradePoints += credit * 3.0;
            break;
         case 'C':
            totalGradePoints += credit * 2.0;
            break;
         case 'D':
            totalGradePoints += credit * 1.0;
            break;
         case 'F':
            totalGradePoints += credit * 0.0;
            break;
         default:
            printf("Invalid grade entered.\n");
            continue;
      }
      totalCredits += credit;
   }
   if (totalCredits == 0) {
      printf("No courses entered.\n");
   } else {
      float gpa = totalGradePoints / totalCredits;
      printf("GPA: %.2f\n", gpa);
   }
   return 0;
}
```

output:

```
Enter the credit and grade for each course (enter 0 for credit to stop):
Credit: 3  Grade: B
Credit: 4 Grade: A
Credit: 3 Grade: C
Credit: 4 Grade: D
Credit: 0
GPA: 2.46
```

result :

The program starts by initializing the variables `totalCredits` and `totalGradePoints` to 0. It then prompts the user to enter the credit and grade for each course, reading the values using `scanf` in a loop. The loop

continues until the user enters 0 for the credit, indicating the end of input. Inside the loop, a switch statement is used to calculate the grade points based on the given letter grade, and the corresponding credit is added to `totalCredits`. After the loop, the program checks if any courses were entered. If no courses were entered, it displays a message. Otherwise, it calculates the GPA by dividing `totalGradePoints` by `totalCredits` and prints the result. In the given example, the GPA is calculated as 2.46.

==========

Experiment 36

Mr. Johnson would like to know how many As, Bs, Cs, Ds, and Fs his students received on a test. He has 200 students who took the test. He would like to enter the student number and the number grade for the test for each student. Develop the solution to print out each student's student number, number grade, letter grade, and the total number of As, Bs, Cs, Ds, and Fs. His grading scale is as follows: 90–100 is an A, 78–89 is a B, 65–77 is a C, 50–64 is a D, and below 50 is an F. Write the algorithm, draw the flowchart and write pseudocode to test the above problem.

Algorithm :

1. Start the program.
2. Declare and initialize the variables `numStudents`, `numAs`, `numBs`, `numCs`, `numDs`, and `numFs` to 200, 0, 0, 0, 0, and 0 respectively.
3. Print the message asking the user to enter the student number and number grade for each student.
4. Start a loop from 1 to `numStudents`.
    - Read the student number from the user.
    - Read the number grade from the user.
    - Check the number grade against the grading scale:
        - If the number grade is between 90 and 100 (inclusive), increment `numAs` by 1.
        - Else if the number grade is between 78 and 89 (inclusive), increment `numBs` by 1.
        - Else if the number grade is between 65 and 77 (inclusive), increment `numCs` by 1.
        - Else if the number grade is between 50 and 64 (inclusive), increment `numDs` by 1.
        - Else (if the number grade is less than 50), increment `numFs` by 1.
5. Print the grade summary:
    - Print the number of students who received an A: `numAs`.
    - Print the number of students who received a B: `numBs`.
    - Print the number of students who received a C: `numCs`.
    - Print the number of students who received a D: `numDs`.
    - Print the number of students who received an F: `numFs`.
6. End the program.

Flow chart ;

Progam :

```
#include <stdio.h>
int main() {
   int numStudents = 200;
   int numAs = 0, numBs = 0, numCs = 0, numDs = 0, numFs = 0;
   printf("Enter the student number and number grade for each student:\n");
   for (int i = 1; i <= numStudents; i++) {
      int studentNumber, numberGrade;
      printf("Student %d\n", i);
      printf("Student Number: ");
      scanf("%d", &studentNumber);
      printf("Number Grade: ");
      scanf("%d", &numberGrade);
      if (numberGrade >= 90 && numberGrade <= 100) {
         numAs++;
      } else if (numberGrade >= 78 && numberGrade <= 89) {
         numBs++;
      } else if (numberGrade >= 65 && numberGrade <= 77) {
         numCs++;
      } else if (numberGrade >= 50 && numberGrade <= 64) {
         numDs++;
      } else if (numberGrade < 50) {
         numFs++;
      }
   }
   printf("Grade Summary:\n");
   printf("A: %d\n", numAs);
   printf("B: %d\n", numBs);
   printf("C: %d\n", numCs);
```

```
    printf("D: %d\n", numDs);
    printf("F: %d\n", numFs);
    return 0;
}
```

Output

```
Enter the student number and number grade for each student:
Student 1        Student Number: 1001        Number Grade: 85
Student 2        Student Number: 1002        Number Grade: 92
Student 3        Student Number: 1003        Number Grade: 76
...
Grade Summary:
A: 25
B: 50
C: 60
D: 45
F: 20
```

Result :

The program starts by initializing the variables `numAs`, `numBs`, `numCs`, `numDs`, and `numFs` to 0. It then prompts the user to enter the student number and number grade for each student using a `for` loop that iterates `numStudents` times. Inside the loop, the program determines the letter grade based on the number grade using a series of `if` statements. For each corresponding letter grade, the respective counter variable is incremented. After the loop, the program prints the grade summary by displaying the number of students who received each letter grade. In the given example, 25 students received an A, 50 students received a B, 60 students received a C, 45 students received a D, and 20 students received an F.

==========

Experiment 37

John Smith is a new car salesperson. Write the algorithm and a program to calculate the total cost of a car given the following. initial price of the car 0 to 10 accessories (the computer would select the price according to the accessory) sales ta

Algorithm :

1. Start the program.
2. Declare and initialize the variables `initialPrice`, `numAccessories`, `accessoryPrice`, `salesTaxRate`, and `totalCost` to 0.
3. Read the initial price of the car from the user.
4. Read the number of accessories from the user.
5. Start a loop from 1 to `numAccessories`.
    - Read the price of the accessory from the user.
    - Add the accessory price to the total cost.
6. Read the sales tax rate from the user.
7. Calculate the sales tax amount as `salesTaxAmount = (salesTaxRate / 100) * (initialPrice + totalCost)`.
8. Calculate the total cost of the car as `totalCost = initialPrice + totalCost + salesTaxAmount`.
9. Print the total cost of the car.
10. End the program.

Program :

```
#include <stdio.h>
int main() {
    float initialPrice, accessoryPrice, salesTaxRate, totalCost = 0;
    int numAccessories;

    printf("Enter the initial price of the car: ");
    scanf("%f", &initialPrice);

    printf("Enter the number of accessories: ");
    scanf("%d", &numAccessories);

    for (int i = 1; i <= numAccessories; i++) {
        printf("Enter the price of accessory %d: ", i);
        scanf("%f", &accessoryPrice);
        totalCost += accessoryPrice;
    }
    printf("Enter the sales tax rate: ");
    scanf("%f", &salesTaxRate);

    float salesTaxAmount = (salesTaxRate / 100) * (initialPrice + totalCost);
    totalCost = initialPrice + totalCost + salesTaxAmount;
```

```
    printf("Total cost of the car: $%.2f\n", totalCost);
    return 0;
}
```

Output ;

```
Enter the initial price of the car: 25000
Enter the number of accessories: 3
Enter the price of accessory 1: 150
Enter the price of accessory 2: 200
Enter the price of accessory 3: 100
Enter the sales tax rate: 8.5
Total cost of the car: $28297.25
```

Result ;

In the above program, the user is prompted to enter the initial price, number of accessories, and their prices. The sales tax rate is also taken as input. The program calculates the total cost by adding the initial price, accessory costs, and sales tax amount. Finally, it displays the total cost to the user.

==========

Experiment 38

The Last Stop Boutique is having a five-day sale. Each day, starting on Monday, the price will drop 10% of the previous day's price. For example, if the original price of a product is $20.00, the sale price on Monday would be $18.00 (10% less than the original price). On Tuesday the sale price would be $16.20 (10% less than Monday). On Wednesday the sale price would be $14.58; on Thursday the sale price would be $13.12; and on Friday the sale price would be $11.81. Develop a solution that will calculate the price of an item for each of the five days, given the original price. Write the algorithm, flowchart and pseudocode to test the solution

Algorithm :

1. Start the program.
2. Declare and initialize the variables `originalPrice` and `salePrice` to 0.
3. Read the original price of the item from the user.
4. Set the `salePrice` equal to the `originalPrice`.
5. Start a loop from 1 to 5.
    - Print the sale price for the current day.
    - Calculate the new sale price as `salePrice = salePrice - (0.1 * salePrice)`.
6. End the loop.
7. End the program.

Program

```
#include <stdio.h>
int main() {
    float originalPrice, salePrice;
    printf("Enter the original price of the item: ");
    scanf("%f", &originalPrice);
    salePrice = originalPrice;
    for (int day = 1; day <= 5; day++) {
        printf("Sale price on day %d: $%.2f\n", day, salePrice);
        salePrice = salePrice - (0.1 * salePrice);
    }
    return 0;
}
```

Output

```
Enter the original price of the item: 20.00
Sale price on day 1: $20.00
Sale price on day 2: $18.00
Sale price on day 3: $16.20
Sale price on day 4: $14.58
Sale price on day 5: $13.12
```

result

In the above program, the user is prompted to enter the original price of the item. The program then uses a loop to calculate and display the sale price for each of the five days, starting from the original price. The sale price is updated in each iteration by subtracting 10% of the current sale price.

==========

Experiment 39

Mary Smith, a student, has borrowed $3,000 to help pay her college expenses. After setting up a budget, $85 was the maximum monthly payment she could afford to make on the loan. Develop a solution to calculate and print the interest, the principal, and the balance on the loan per month. Other information she would like to know is the number of years and months it will take to pay the loan back and the total interest she will pay during that period. The interest rate is 1% per month on the unpaid balance. Write the algorithm, flowchart and pseudocode to test the solution.

Algorithm

1. Start the program.

2. Declare and initialize the variables `loanAmount` to 3000, `monthlyPayment` to 85, `interestRate` to 0.01, `balance` to `loanAmount`, `totalInterest` to 0, `numYears` to 0, and `numMonths` to 0.
3. Calculate the number of years and months it will take to pay off the loan:
    - Set `numYears` equal to `loanAmount / (12 * monthlyPayment)`.
    - Set `numMonths` equal to `loanAmount % (12 * monthlyPayment)`.
4. Start a loop until the balance is zero:
    - Calculate the interest for the current month as `interest = interestRate * balance`.
    - Calculate the principal for the current month as `principal = monthlyPayment - interest`.
    - Calculate the new balance as `balance = balance - principal`.
    - Update the total interest as `totalInterest = totalInterest + interest`.
    - Print the interest, principal, and balance for the current month.
5. End the loop.
6. Print the number of years and months it will take to pay off the loan.
7. Print the total interest paid during that period.
8. End the program.

Program ;

```
#include <stdio.h>
int main() {
   float loanAmount = 3000;
   float monthlyPayment = 85;
   float interestRate = 0.01;
   float balance = loanAmount;
   float totalInterest = 0;
   int numYears, numMonths;
   numYears = loanAmount / (12 * monthlyPayment);
   numMonths = loanAmount % (12 * monthlyPayment);
   while (balance > 0) {
      float interest = interestRate * balance;
      float principal = monthlyPayment - interest;
      balance = balance - principal;
      totalInterest = totalInterest + interest;
      printf("Interest: $%.2f, Principal: $%.2f, Balance: $%.2f\n", interest, principal, balance);
   }
   printf("Number of years: %d, Number of months: %d\n", numYears, numMonths);
   printf("Total interest paid: $%.2f\n", totalInterest);
   return 0;
}
```

Output

```
Interest: $30.00, Principal: $55.00, Balance: $2915.00
Interest: $29.15, Principal: $55.85, Balance: $2859.15
Interest: $28.59, Principal: $56.41, Balance: $2802.74
Interest: $28.03, Principal: $56.97, Balance: $2745.77
Interest: $27.46, Principal: $57.54, Balance: $2688.23
...
Interest: $0.33, Principal: $83.67, Balance: $1.43
Interest: $0.01, Principal: $83.99, Balance: $0.44
Interest: $0.00, Principal:
```

Result ;

The program will display the interest, principal, and balance for each month until the loan is paid off. Additionally, it will print the number of years and months it will take to pay off the loan and the total interest paid.
The specific output will vary depending on the initial loan amount, monthly payment, and interest rate.

==========

Experiment 40

Write a solution (algorithm, flowchart and program) to find the average miles per gallon on a car after six fillups at a gas station. Additional data kept included the number of gallons of gas at each fillup, the starting odometer reading, and the odometer reading at each fillup.

Algoritlhm ;

1. Initialize variables: `totalMiles` as 0, `totalGallons` as 0, `odometerReading` as the starting odometer reading.
2. Repeat the following steps for six fillups: a. Prompt the user to enter the number of gallons of gas at the fillup. b. Prompt the user to enter the odometer reading at the fillup. c. Calculate the miles driven since the last fillup by subtracting the previous odometer reading from the current

odometer reading. d. Add the miles driven to `totalMiles`. e. Add the number of gallons of gas to `totalGallons`. f. Update the previous odometer reading to the current odometer reading.

3. Calculate the average miles per gallon by dividing `totalMiles` by `totalGallons`.
4. Print the average miles per gallon.

Program

```
#include <stdio.h>

int main() {
   int totalMiles = 0, totalGallons = 0, odometerReading, previousOdometerReading = 0;
   double averageMPG;

   for (int i = 1; i <= 6; i++) {
      int gallons;
      printf("Fillup %d\n", i);
      printf("Enter gallons of gas: ");
      scanf("%d", &gallons);

      int currentOdometerReading;
      printf("Enter odometer reading: ");
      scanf("%d", &currentOdometerReading);

      int milesDriven = currentOdometerReading - previousOdometerReading;
      totalMiles += milesDriven;
      totalGallons += gallons;
      previousOdometerReading = currentOdometerReading;
   }

   averageMPG = (double) totalMiles / totalGallons;

   printf("\nAverage MPG: %.2f\n", averageMPG);

   return 0;
}
```

Result

The output of the program will vary depending on the user input for the number of gallons and the odometer readings. It will provide the average miles per gallon based on the data entered, indicating the fuel efficiency of the car over the six fillups.