Galgotias College of Engineering & Technology, Greater Noida

Affiliated to Dr. A. P. J. AKTU, Lucknow

Department of Computer Science & Engineering

LAB MANUAL

PPS Lab, BCS-151/BCS-251

(B. TECH, FIRST YEAR)

Session: -2025-26(Odd/Even)

INSTITUTE VISION & MISSION

Vision of the Institute

To be a globally recognized institution distinguished by excellence in education, research, innovation, and entrepreneurship, producing competent & socially responsible technocrats for sustainable growth.

Mission of the Institute

- To cultivate a student-centric ecosystem that fosters experiential learning, ethical problem-solving, and sustainability.
- To provide a conducive environment for the professional growth of faculty and staff through research and global collaboration, which contributes to the nation's overall development.
- To nurture a culture of active citizenship through excellence in education, entrepreneurship, and innovation, producing socially responsible and competent technocrats.

DEPARTMENT VISION, MISSION & PEOS

Department Vision

To be a center of excellence in delivering innovative education and cutting-edge research, with a strong focus on emerging technologies and industry-aligned solutions.

Department Mission

- **M1:** To foster and effective learning environment through quality teaching practices, encouraging excellence in education and research.
- **M2:** To drive innovation, consultancy, and entrepreneurship by fostering strong, collaborative partnerships between industry and academia.
- **M3:** To prepare students for the corporate world by building self-confidence, professionalism, and interpersonal skills, while promoting sustainability, ethical values, and value-based living.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

- **PEO1:** To empower students to excel in their careers by mastering Computer Science & Engineering with modern tools and emerging technologies.
- **PEO2:** To address real-world community issues through thoughtful evaluation and innovative problem-solving.
- **PEO3:** To cultivate strong communication and teamwork skills for success in industry and research, supported by continuous professional development and lifelong learning.

KNOWLEDGE AND ATTITUDE PROFILE (WK):

Knowledge Profiles (WK)	
No.	**Description**
WK1	A systematic, theory-based understanding of the **natural sciences** applicable to the discipline and awareness of relevant **social sciences.**
WK2	Conceptually-based **mathematics**, numerical analysis, data analysis, statistics and formal aspects of computer and information science to support detailed analysis and modelling applicable to the discipline.
WK3	A systematic, theory-based formulation of **engineering fundamentals** required in the engineering discipline.
WK4	Engineering **specialist knowledge** that provides theoretical frameworks and bodies of knowledge for the accepted practice areas in the engineering discipline; much is at the forefront of the discipline.
WK5	Knowledge, including efficient resource use, environmental impacts, whole-life cost, re-use of resources, net zero carbon, and similar concepts, that supports **engineering design and operations** in a practice area.
WK6	Knowledge of **engineering practice** (technology) in the practice areas in the engineering discipline.
WK7	**Knowledge of** the role of engineering in society and identified issues in engineering practice in the discipline, such as the professional responsibility of an engineer to public safety and **sustainable development***.
WK8	Engagement with selected knowledge in the current **research literature** of the discipline, awareness of the power of critical thinking and creative approaches to evaluate emerging issues.
WK9	**Ethics, inclusive behavior and conduct.** Knowledge of professional ethics, responsibilities, and norms of engineering practice. Awareness of the need for diversity by reason of ethnicity, gender, age, physical ability etc. with mutual understanding and respect, and of inclusive attitudes

PROGRAM OUTCOMES (POs)

Program Outcomes	Statement
PO1	**Engineering Knowledge:** Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems..
PO2	**Problem Analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4).
PO3	**Design/Development of Solutions:** Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)
PO4	**Conduct Investigations of Complex Problems:** Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).
PO5	**Engineering Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)
PO6	**The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).
PO7	**Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)
PO8	**Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams
PO9	**Communication:** Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective

	reports and design documentation, make effective presentations considering cultural, language, and learning differences
PO10	**Project Management and Finance:** Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.
PO11	**Life-Long Learning:** Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change. (WK8)

PROGRAM SPECIFIC OUTCOMES (PSOs)

Program Specific Outcomes (PSO)	**Statement** **Engineering graduates will be able to:**
PSO1	To integrate theoretical foundations of computer science into the design of computing systems, demonstrating insight into trade-offs and essential design factors.
PSO2	To design, implement, and validate software systems tailored for diverse applications, ensuring adherence to real-time performance constraints.

COURSE DESCRIPTION

This course introduces students to the fundamentals of programming using the C language. The course emphasizes structured programming, logical thinking, and efficient coding practices. Students will learn how to formulate algorithms, convert them into structured C programs, and solve real-world computational problems. Key topics include data types, control structures, functions, arrays, pointers, strings, and file handling. By the end of the course, students will be able to design and implement modular, reusable, and efficient code using standard C programming practices. The course includes hands-on programming sessions to reinforce theoretical concepts, encouraging students to build problem-solving skills that will support further courses in data structures, algorithms, and systems programs.

COURSE OUTCOMES:

COs	**On completion of this course, the students will be able to**	**Bloom's Knowledge Level (KL**
BCS251.1	Write algorithms for arithmetic and logical problem.	K1, K2
BCS251.2	Write program using modular concept, array, loop and structure.	K3
BCS251.3	Implement file handling concept using C program.	K3

Mapping of Course Outcome with Program Outcome and Program Specific Outcomes

Slight (low) 2. Moderate (medium) 3. Substantial (high)

PO CO	**PO 1**	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PSO 1**	**PSO 2**
BCS251.1	3	2	-	-	3	-	3	3	-	-	3	-	2
BCS251.2	3	2	3	-	3	-	3	3	-	-	3	-	2
BCS251.3	3	2	3	-	3	-	3	3	-	-	3	-	2
Average	**3**	**2**	**3**	**-**	**3**	**-**	**3**	**3**	**-**	**-**	**3**	-	2

PPS Lab AKTU Syllabus

BCS151 / BCS251: PROGRAMMING FOR PROBLEM SOLVING LAB

1. WAP that accepts the marks of 5 subjects and finds the sum and percentage marks obtained
   by the student.
2. WAP that calculates the Simple Interest and Compound Interest. The Principal, Amount, Rate
   of Interest and Time are entered through the keyboard.
3. WAP to calculate the area and circumference of a circle.
4. WAP that accepts the temperature in Centigrade and converts into Fahrenheit using the formula C/5=(F-32)/9.
5. WAP that swaps values of two variables using a third variable.
6. WAP that checks whether the two numbers entered by the user are equal or not.
7. WAP to find the greatest of three numbers.
8. WAP that finds whether a given number is even or odd.
9. WAP that tells whether a given year is a leap year or not.
10. WAP that accepts marks of five subjects and finds percentage and prints grades according to the following criteria:

    Between 90-100%-----Print 'A'
    80-90%-----------------Print 'B'
    60-80%-----------------Print 'C'
    Below 60%------------Print 'D'
11. WAP that takes two operands and one operator from the user, perform the operation, and prints the result by using Switch statement.
12. WAP to print the sum of all numbers up to a given number.
13. WAP to find the factorial of a given number.
14. WAP to print sum of even and odd numbers from 1 to N numbers.
15. WAP to print the Fibonacci series.
16. WAP to check whether the entered number is prime or not.
17. WAP to find the sum of digits of the entered number.
18. WAP to find the reverse of a number.
19. WAP to print Armstrong numbers from 1 to 100.
20. WAP to convert binary number into decimal number and vice versa.
21. WAP that simply takes elements of the array from the user and finds the sum of these elements.
22. WAP that inputs two arrays and saves sum of corresponding elements of these arrays in a third array and prints them.
23. WAP to find the minimum and maximum element of the array.
24. WAP to search an element in a array using Linear Search.
25. WAP to sort the elements of the array in ascending order using Bubble Sort technique.

26. WAP to add and multiply two matrices of order nxn.
27. WAP that finds the sum of diagonal elements of a mxn matrix.
28. WAP to implement strlen (), strcat (),strcpy () using the concept of Functions.
29. Define a structure data type TRAIN_INFO. The type contain Train No.: integer type Train name: string Departure Time: aggregate type TIME Arrival Time: aggregate type TIME Start station: string End station: string The structure type Time contains two integer members: hour and minute. Maintain a train timetable and implement the following operations:
    a. List all the trains (sorted according to train number) that depart from a particular section.
    b. List all the trains that depart from a particular station at a particular time.
    c. List all he trains that depart from a particular station within the next one hour of a given time.
    d. List all the trains between a pair of start station and end station.
30. WAP to swap two elements using the concept of pointers.
31. WAP to compare the contents of two files and determine whether they are same or not.
32. WAP to check whether a given word exists in a file or not. If yes then find the number of times it occurs.

Note: ***a) The Instructor may add/delete/modify/tune experiments, wherever he/she feels in a justified manner.***

b) The subject teachers are suggested to use the concept of project based learning. The subject teacher may giver certain use cases/case studies where student is able to apply multiple concepts in one single program.

c) It is also suggested that open source tools should be preferred to conduct the lab. Some open source online compiler to conduct the C lab are as follows:

https://www.jdoodle.com/c-online-compiler/

https://www.tutorialspoint.com/compile_c_online.php https://www.programiz.com/c-programming/online-compiler/

https://www.hackerrank.com/

List of Experiments(to be conducted):

S. No.	Object / Aim of the Experiment	As per the AKTU Syllabus / Value-added	Mapped with CO_X	Major Equipments used.
1	WAP that accepts the marks of 5 subjects and finds the sum and percentage marks obtained by the student	AKTU	CO1	Turbo c, vs code, dev c++
2	WAP that calculates the Simple Interest and Compound Interest. The Principal, Amount, Rate of Interest and Time are entered through the keyboard.	AKTU	CO1	Turbo c, vs code, dev c++
3	WAP that swaps values of two variables using a third variable.	AKTU	CO1	Turbo c, vs code, dev c++
4	WAP that checks whether the two numbers entered by the user are equal or not.	AKTU	CO1	Turbo c, vs code, dev c++
5	WAP to find the greatest of three numbers.	AKTU	CO1	Turbo c, vs code, dev c++
6	WAP that finds whether a given number is even or odd.	AKTU	CO1	Turbo c, vs code, dev c++
7	WAP that tells whether a given year is a leap year or not.	AKTU	CO1	Turbo c, vs code, dev c++
8	WAP that accepts marks of five subjects and finds percentage and prints grades according to the following criteria: Between 90-100%-----Print ‘A’ 80-90%-----------------Print ‘B’	AKTU	CO2	Turbo c, vs code, dev c++

	60-80%-----------------Print 'C' Below 60%------------Print 'D'			
9	WAP that takes two operands and one operator from the user, perform the operation, and prints the result by using Switch statement.	AKTU	CO2	Turbo c, vs code, dev c++
10	WAP to print the sum of all numbers up to a given number.	AKTU	CO2	Turbo c, vs code, dev c++
11	WAP to find the factorial of a given number.	AKTU	CO2	Turbo c, vs code, dev c++
12	WAP to print sum of even and odd numbers from 1 to N numbers.	AKTU	CO2	Turbo c, vs code, dev c++
13	WAP to print the Fibonacci series.	AKTU	CO2	Turbo c, vs code, dev c++
14	WAP to check whether the entered number is prime or not.	AKTU	CO2	Turbo c, vs code, dev c++
15	WAP to find the sum of digits of the entered number.	AKTU	CO2	Turbo c, vs code, dev c++
16	WAP to find the reverse of a number.	AKTU	CO2	Turbo c, vs code, dev c++
17	WAP to print Armstrong numbers from 1 to 100.	AKTU	CO2	Turbo c, vs code, dev c++

18	WAP to convert binary number into decimal number and vice versa.	AKTU	CO2	Turbo c, vs code, dev c++
19	WAP that simply takes elements of the array from the user and finds the sum of these elements.	AKTU	CO2	Turbo c, vs code, dev c++
20	WAP that inputs two arrays and saves sum of corresponding elements of these arrays in a third array and prints them.	AKTU	CO2	Turbo c, vs code, dev c++
21	WAP to find the minimum and maximum element of the array.	AKTU	CO2	Turbo c, vs code, dev c++
22	WAP to search an element in a array using Linear Search.	AKTU	CO2	Turbo c, vs code, dev c++
23	WAP to sort the elements of the array in ascending order using Bubble Sort technique.	AKTU	CO2	Turbo c, vs code, dev c++
24	WAP to sort the elements in ascending order using selection sort and insertion sort.	VALUE ADDED	CO2	Turbo c, vs code, dev c++
25	Write a program to search an element in an array using Binary Search.	VALUE ADDED	CO2	Turbo c, vs code, dev c++
26	WAP to add and multiply two matrices of order nxn.	AKTU	CO2	Turbo c, vs code, dev c++
27	WAP that finds the sum of diagonal elements of a mxn matrix.	AKTU	CO2	Turbo c, vs code, dev c++

28	WAP to implement strlen (), strcat (),strcpy () using the concept of Functions.	AKTU	CO2	Turbo c, vs code, dev c++
29	Define a structure data type TRAIN_INFO. The type contain Train No.: integer typeTrain name: string Departure Time: aggregate type TIME Arrival Time: aggregate typeTIME Start station: string End station: string The structure type Time contains twointeger members: hour and minute. Maintain a train timetable and implement the following operations: a. List all the trains (sorted according to train number) that depart from a particularsection. b. List all the trains that depart from a particular station at a particular time. c. List all he trains that depart from a particular station within the next one hour of a given time. d. List all the trains between a pair of start station and end station.	AKTU	CO2	Turbo c, vs code, dev c++
30	WAP to swap two elements using the concept of pointers.	AKTU	CO2	Turbo c, vs code, dev c++
31	WAP to compare the contents of two files and determine whether they are same or not.	AKTU	CO3	Turbo c, vs code, dev c++
32	WAP to check whether a given word exists in a file or not. If yes then find the number of times it occurs.	AKTU	CO3	Turbo c, vs code, dev c++
33	Define a structure data type name STUDENT. The type contains student id: string type, student name: type string, student lage:type integer, total marks: float type. Display all the record of the student.	VALUE ADDED	CO2	Turbo c, vs code, dev c++
34	Define a structure data type name STUDENTS. The type contains student id: string type, student name: ype string, student	VALUE ADDED	CO2	Turbo c, vs code, dev c++

35	Create structure data type name ADDRESS;- The type contains city: string type, pincode: type integer. Create structure data 3 type name EMPLOYEE;- The type contains name: string type. Display allthe information of the Employee using concept of nested structure	VALUE ADDED	CO2	Turbo c, vs code, dev c++
36	Write a program to compare the contents of two files and determine whether they are the same or not.	VALUE ADDED	CO3	Turbo c, vs code, dev c++
37	WAP to check whether the given no is Palindrom or not.	VALUE ADDED	CO2	Turbo c, vs code, dev c++
38	WAP to check whether the no is Perffect or not.	VALUE ADDED	CO2	Turbo c, vs code, dev c++
39	WAP to make a hotel menu using switch case.	VALUE ADDED	CO2	Turbo c, vs code, dev c++
40	WAP to sum two numbers without using '+' operator.	VALUE ADDED	CO2	Turbo c, vs code, dev c++
41	WAP to print Right and Left pyramid pattern.	VALUE ADDED	CO2	Turbo c, vs code, dev c++
42	WAP to find total no of alphabtes, digits and specail characters in a string.	VALUE ADDED	CO2	Turbo c, vs code, dev c++

Experiment No. 1

Objective / Aim: Write a program that accepts the marks of 5 subjects and finds the sum and percentage of marks obtained by the student.

Learning Outcomes (LOs):

LO1: Apply basic input and output operations in C.

LO2: Perform arithmetic calculations to compute the total and percentage.

LO3: Develop problem-solving skills using variables and expressions.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

sum=m1+m2+m3+m4+m5

per=(sum/500)*100

Algorithm:

```
Step 1: Read five subject marks and store them into variables.
Step 2: Calculate sum of all subjects and store in total = eng + phy + chem + math + comp.
Step 3: Divide sum of all subjects by total number of subject to find average i.e.    average = total / 5.
Step 4: Calculate percentage using percentage = (total / 500) * 100.
Step 5: Finally, print resultant values total, average and percentage.
```

Flowchart:

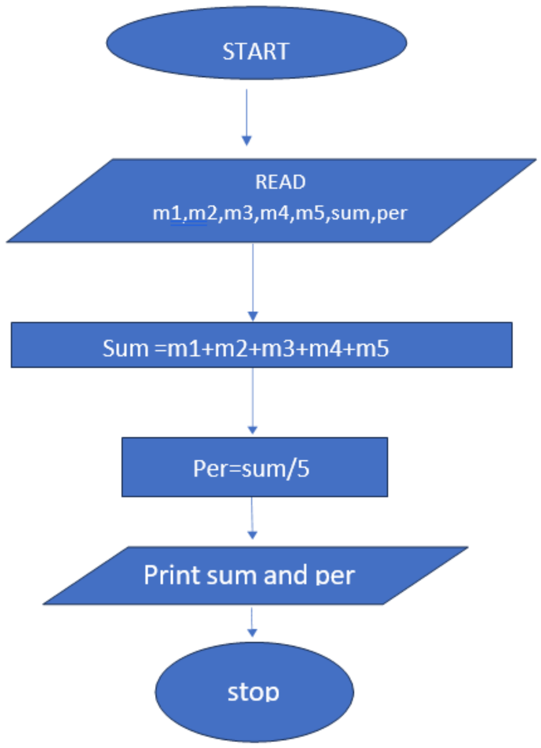
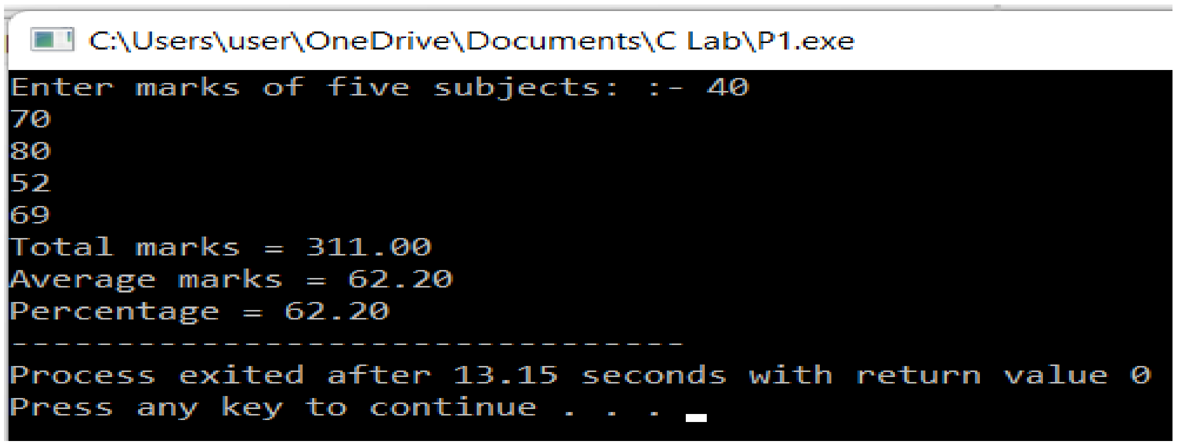


Fig.1.1 Flowchart to print the sum and average of marks.

Input/Output(Screenshot):



```
Enter marks of five subjects: :- 40
70
80
52
69
Total marks = 311.00
Average marks = 62.20
Percentage = 62.20
--------------------------------
Process exited after 13.15 seconds with return value 0
Press any key to continue . . .
```

Fig. 1.2 Input/Output screenshot of the total sum, percentage and average of the given marks.

Precautions and Errors:

- follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

1.1 What is the use of printf() and scanf() functions? Also explain format specifiers

1.2 What is a built-in function in C?

Experiment No. 2

Objective / Aim: Write a program that calculates the Simple Interest and Compound Interest. The Principal Amount, Rate of Interest, and Time are entered through the keyboard.

Learning Outcomes (LOs):

LO1: Understand and implement formulas for financial computations in C.

LO2: Take multiple inputs and perform mathematical operations.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

$$\text{Simple Interest} = (P*r*t)/100$$

$$A=P*(1+r/100)^{t}$$

$$\text{Compound Interest (C.I.)}=\text{Amount (A)}-\text{Principal (P)}$$

Algorithm:

```
Step 1: Start
Step 2: Input Principal Amount (P), Rate of Interest (R), and Time Period (T).
Step 3: Calculate Simple Interest (SI) SI = (P * R * T) / 100
Step 4: Calculate Compound Interest (CI) CI = P * (1 + R/100)^T - P
Step 5: Display SI and CI
Step 6: End
```

Flowchart:

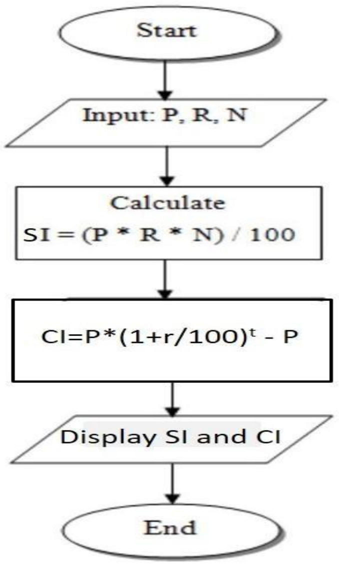
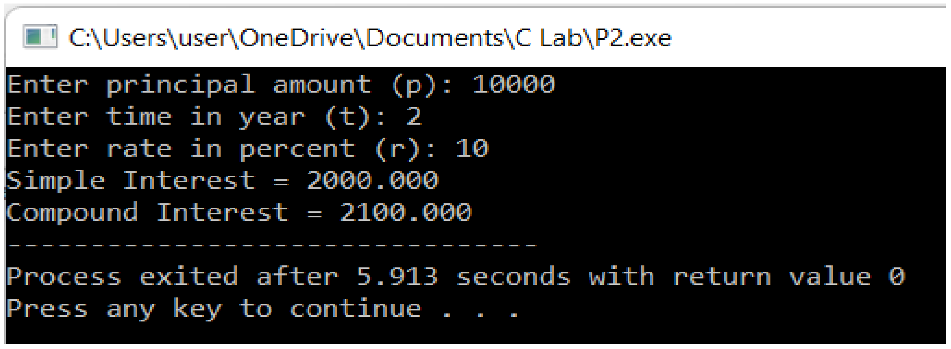


Fig. 2.1 Flowchart to find SI and CI.

Input/Output(Screenshot):



```
Enter principal amount (p): 10000
Enter time in year (t): 2
Enter rate in percent (r): 10
Simple Interest = 2000.000
Compound Interest = 2100.000
--------------------------------
Process exited after 5.913 seconds with return value 0
Press any key to continue . . .
```

Fig. 2.2 Fig. 1.2 Input/Output screenshot of displaying SI and CI.

Precautions and Errors:

- follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

2.1 What is a Preprocessor?

2.2 What is variable initialization and why is it important?

Experiment No. 3

Objective: Write a program that swaps the values of two variables using a third variable & without a third variable.

Learning Outcomes (LOs):

LO1: Understand the concept of variables and data storage.

LO2: Implement swapping logic using a temporary variable.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

To swap two numbers, use a temporary variable i.e. third variable. First, we assign the value of first variable to temporary variable, then assign the value of second variable to first variable and in the last assign the value of temporary variable to second variable, which is the value of first variable.

Algorithm:

```
Step 1: START
Step2: ENTER x, y
Step3: PRINT x, y
Step4: t = x
Step5: x= y
Step6: y= t
Step7: PRINT x, y
Step8: x=x+y
Step9: y=x-y
Step 10: x=x-y
Step 11: PRINY x, y
Step 12: END
```

Flowchart:

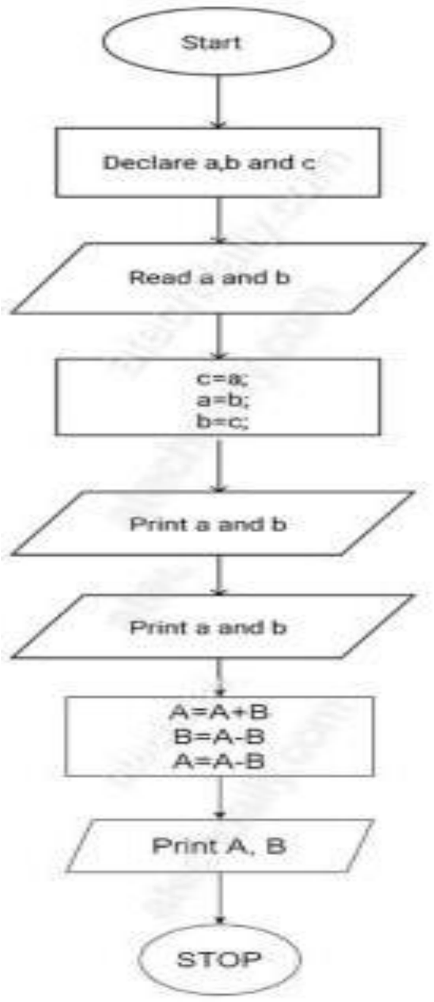
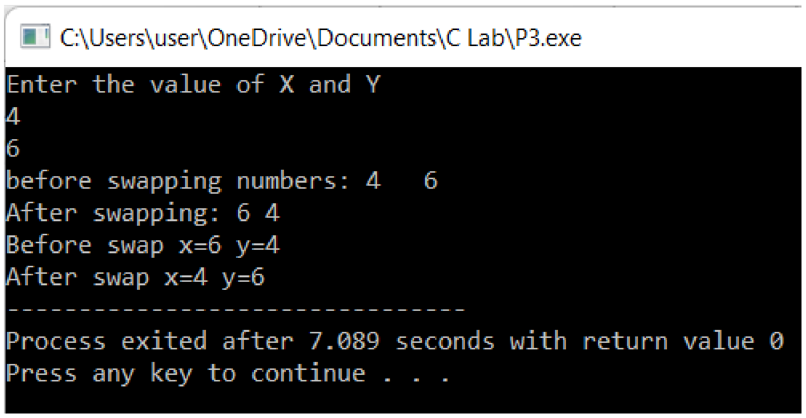


Fig. 3.1 Flowchart to swap to two numbers.

Input/Output (Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\P3.exe
Enter the value of X and Y
4
6
before swapping numbers: 4   6
After swapping: 6 4
Before swap x=6 y=4
After swap x=4 y=6
--------------------------------
Process exited after 7.089 seconds with return value 0
Press any key to continue . . .
```

Fig. 3.2 Input/Output screenshot of swapping two numbers.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement**.**

Questions for Viva-voice:

3.1 What are header files and what are their uses in C programming?

3.2 What do you mean by swapping of two numbers?

Experiment No. 4

Objective / Aim: Write a program that checks whether the two numbers entered by the user are equal or not.

Learning Outcomes (LOs):

LO1: Learn to use relational operators for comparison.

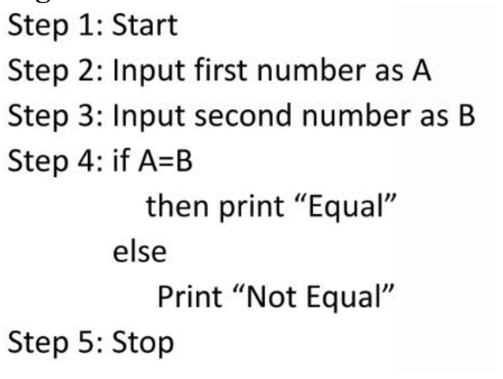
LO2: Implement conditional statements to test equality.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

In this program conditional statement i.e. if else statement. The conditional statements (also known as decision control structures) such as if, if else, switch, etc. are used for decision-making purposes in C programs. They are also known as Decision-Making Statements and are used to evaluate one or more conditions and make the decision whether to execute a set of statements or not. These decision-making statements in programming languages decide the direction of the flow of program execution.

Algorithm:



```
Step 1: Start
Step 2: Input first number as A
Step 3: Input second number as B
Step 4: if A=B
            then print “Equal”
        else
            Print “Not Equal”
Step 5: Stop
```

Flowchart:

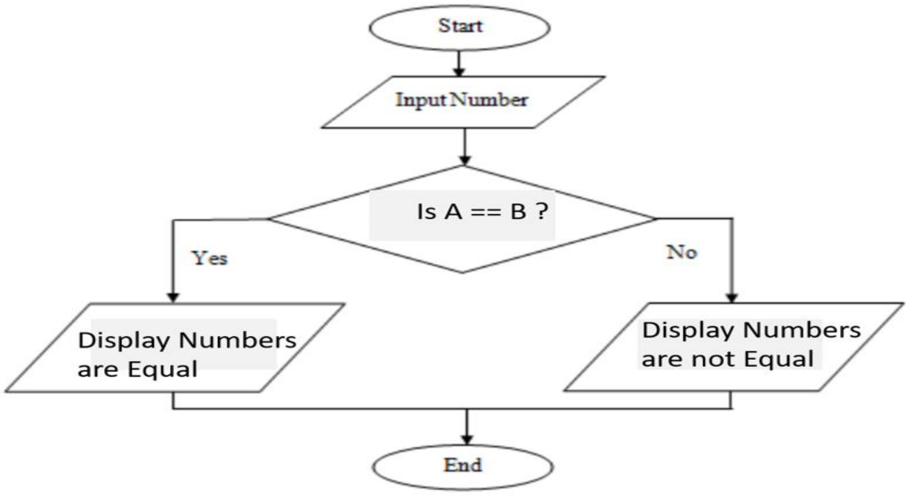
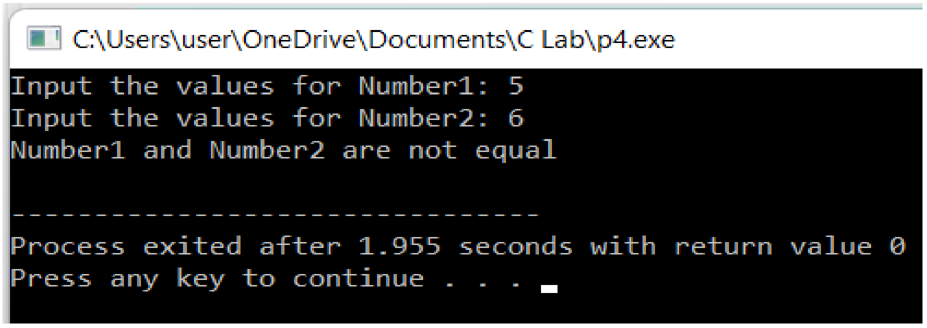


Fig. 4.1 Flowchart to check if two numbers are equal or not.

Input/Output(Screenshot):



```
Input the values for Number1: 5
Input the values for Number2: 6
Number1 and Number2 are not equal

--------------------------------
Process exited after 1.955 seconds with return value 0
Press any key to continue . . .
```

Fig. 4.2 Input/Output screenshot of displaying two numbers. are equal or not .

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

4.1 What is Conditional Statement or decision-making statement?

4.2 Differentiate between = and == operator

Experiment No. 5

Objective / Aim: WAP to find the greatest of three numbers.

Learning Outcomes (LOs):

LO1: Understand nested conditional statements and logical operators.

LO2: Implement decision-making for multiple inputs.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

In this program conditional statement i.e. if, else if, else statement. The conditional statements (also known as decision control structures) such as if, if else, switch, etc. are used for decision-making purposes in C programs. They are also known as Decision-Making Statements and are used to evaluate one or more conditions and make the decision whether to execute a set of statements or not. These decision-making statements in programming languages decide the direction of the flow of program execution.

Algorithm:

```
Step 1. Start
Step 2. Read the three numbers to be compared, as A, B and C
Step 3. Check if A is greater than B.
        3.1 If true, then check if A is greater than C
            If true, print 'A' as the greatest number
            If false, print 'C' as the greatest number
        3.2 If false, then check if B is greater than C
            If true, print 'B' as the greatest number
            If false, print 'C' as the greatest number
Step 4. End
```

Flowchart:

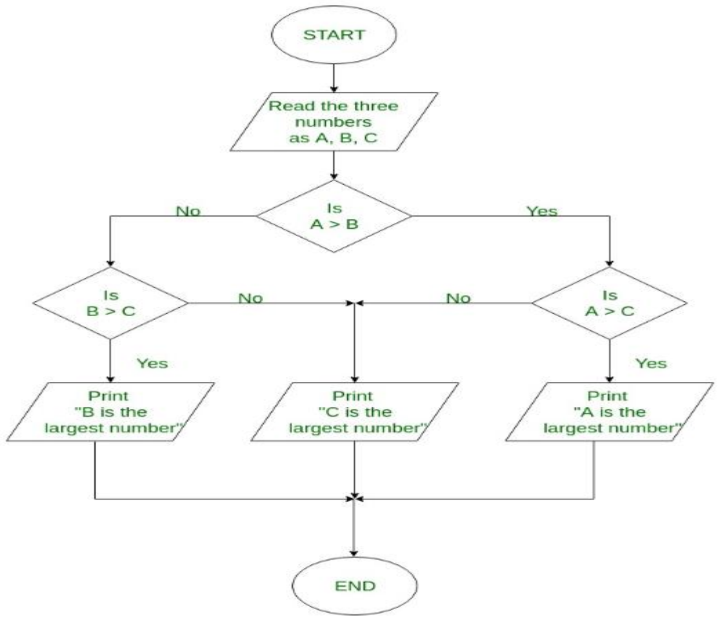
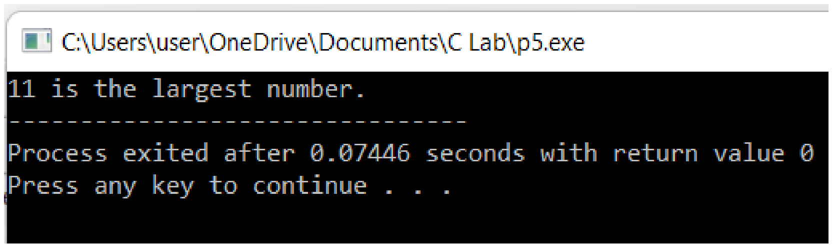

START
Read the three numbers as A, B, C
Is A > B
No
Yes
Is B > C
No
No
Is A > C
Yes
Yes
Print "B is the largest number"
Print "C is the largest number"
Print "A is the largest number"
END

Fig. 5.1 Flowchart to print the largest of three numbers

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p5.exe
11 is the largest number.
--------------------------------
Process exited after 0.07446 seconds with return value 0
Press any key to continue . . .
```

Fig. 5.2 Input/ Output screenshot of displaying the largest of three numbers.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

5.1 What is the use of if else statement?

5.2 What is logical operator?

Experiment No. 6

Objective / Aim: WAP that finds whether a given number is even or odd.

Learning Outcomes (LOs):

LO1: Understand the concept of modulo operations in C.

LO2: Implement conditional statements for checking number properties.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

In this program conditional statement i.e. if else statement. These statements in programming languages decide the direction of the flow of program execution. A number that is completely divisible by 2 is an even number and a number that is not completely divisible by 2 leaving a non-zero remainder is an odd number.

Algorithm:

```
Step 1- Start
Step 2- Read/input the number.
Step 3- if n%2==0 then Display number is even.
Step 4- else Display number is odd.
Step 5- Stop
```

Flowchart:

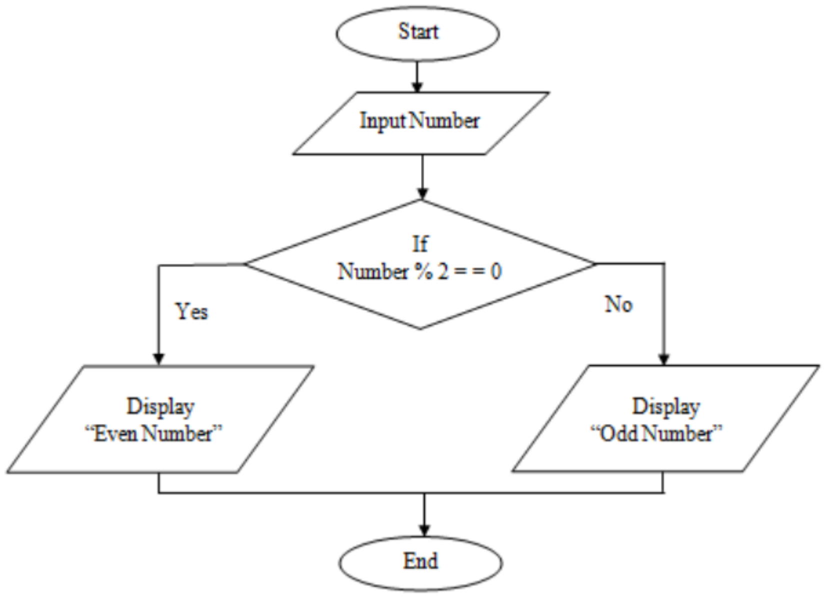
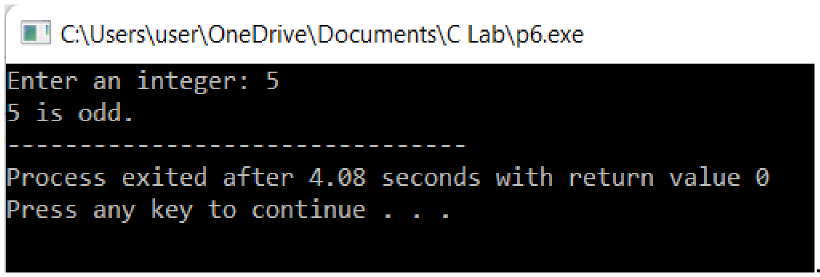


Fig. 6.1 Flowchart to display Even and Odd numbers.

Input/Output(Screenshot):



```
Enter an integer: 5
5 is odd.
--------------------------------
Process exited after 4.08 seconds with return value 0
Press any key to continue . . .
```

Fig. 6.2 Input/ Output screenshot of displaying the number is Even or Odd.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

6.1 Explain relational operator.
6.2 Explain arithmetic operator.

Experiment No. 7

Objective / Aim: WAP that tells whether a given year is a leap year or not.

Learning Outcomes (LOs):

LO1: Understand the logic and conditions for leap year calculation.

LO2: Use nested conditions effectively in C.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

In this program conditional statement i.e. if else statement. These statements in programming languages decide the direction of the flow of program execution.
A year is a leap year if "any one of " the following conditions are satisfied:

1. The year is multiple of 400.
2. The year is a multiple of 4 and not a multiple of 100.

Algorithm:

```
Step 1: Start

Step 2: Take the integer variable year

Step 3: Assign value to the variable

Step 4: Check if year is divisible by 4 but not 100, DISPLAY "leap year"

Step 5: Check if year is divisible by 400, DISPLAY "leap year"

Step 6: Otherwise, DISPLAY "not leap year"

Step 7: Stop
```

Flowchart:

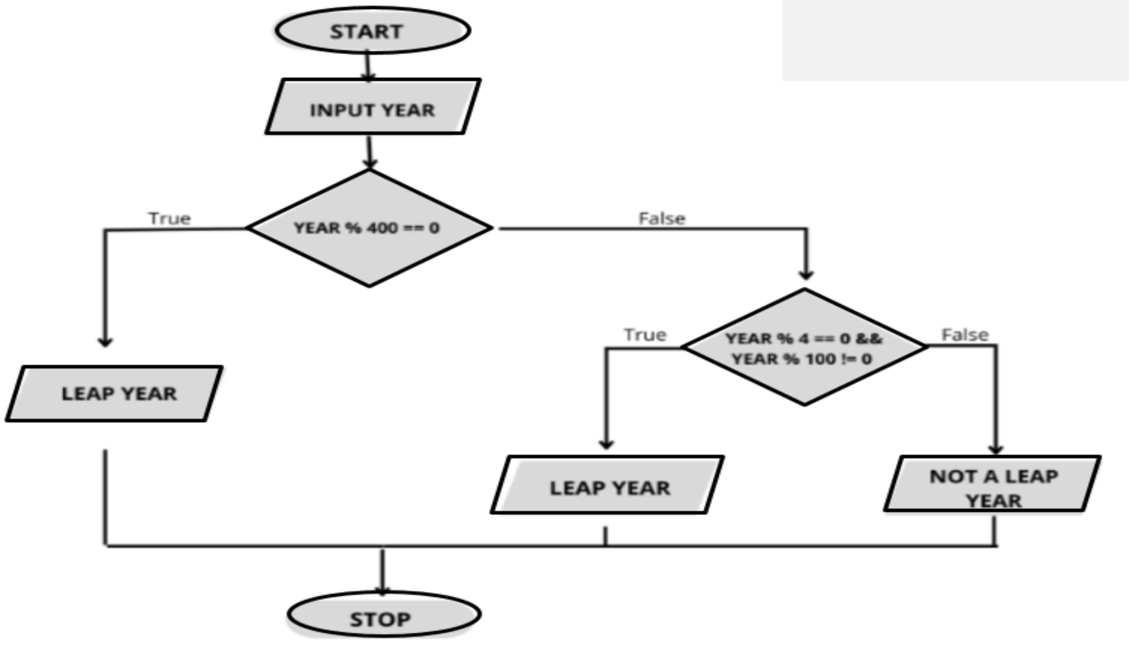
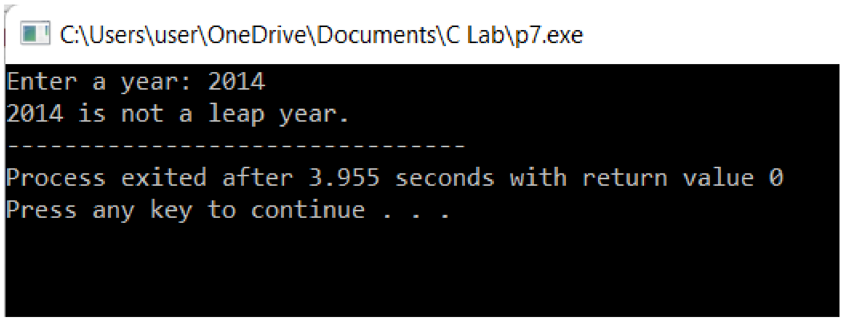


Fig. 7.1 Flowchart to print the given year is leap or not.

Input/Output(Screenshot):



```
Enter a year: 2014
2014 is not a leap year.
--------------------------------
Process exited after 3.955 seconds with return value 0
Press any key to continue . . .
```

Fig. 7.2 Input/ Output screenshot of displaying a given year is Leap or not.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

7.1 What is Conditional Operator?

7.2 What is ternary operator?

Experiment no. 8

Objective/Aim: WAP that accepts marks of five subjects and finds percentage and prints grades according to the following criteria:Between 90-100%--------------Print 'A' 80-90%----------------------------Print 'B' 60-80%---------------------------Print 'C' Below 60%----------------------Print 'D'.

Learning Outcomes (LOs):

LO1: Apply percentage calculation for academic evaluation.

LO2: Implement multiple conditional statements for grading logic.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory:

- The **conditional statements** (also known as branching control structures) such as if, if else, switch, etc. are used for decision-making purposes in C programs.
- They are also known as Decision-Making Statements and are used to evaluate one or more conditions and make the decision whether to execute a set of statements or not. These decision-making statements in programming languages decide the direction of the flow of program execution.

In this program, user will input marks of 5 subjects and calculate percentage using arithmetic operators. Now, for giving grades according to the percentage, we will use if-else-if conditional statements. A particular conditional statement will be executed if true else the control will be transfer to another conditional statement.

Formula: sum=m1+m2+m3+m4+m5;

per=(sum/500)*100;

Algorithm:

```
Step 1: start
Step 2:  read m,m2,m3,m4,m5,per
Step 3:  per=((m1+m2+m3+m4+m5)*100)/500);
Step 4:  if per>=90
            print secured % with grade A
Step 5: else if per>=80 and per<90
```

```
        print grade B
Step 6: else if per>=60 and per<80
        print grade C
Step 7: else
        print grade D
Step 8:  stop
```

Flowchart:

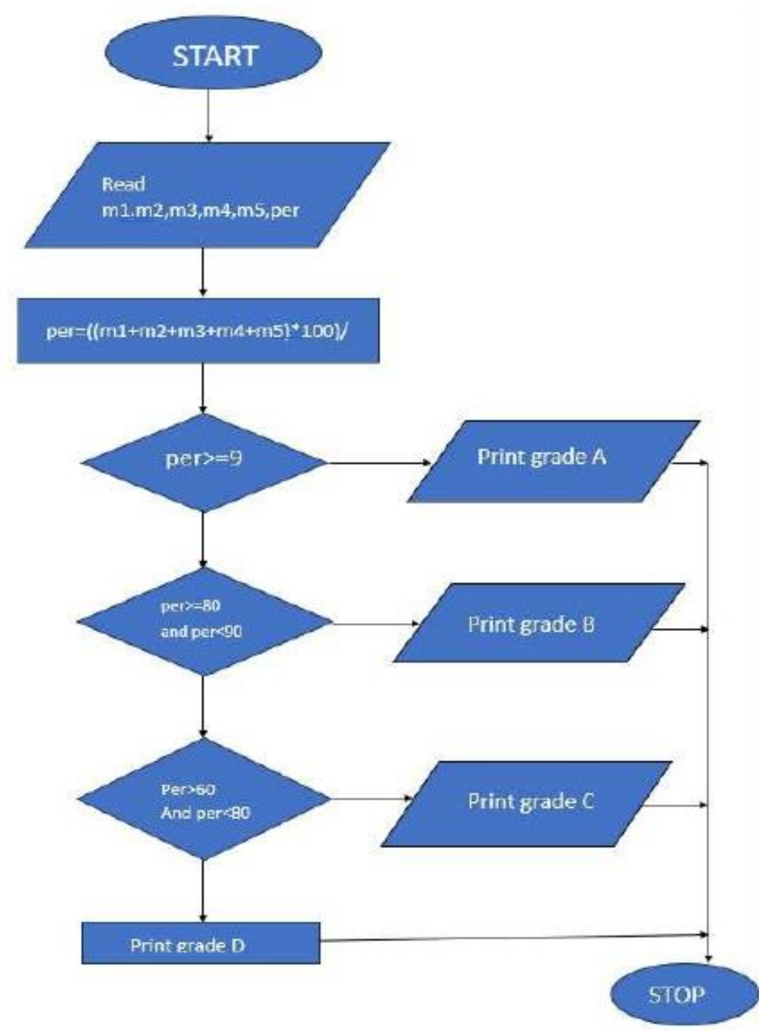
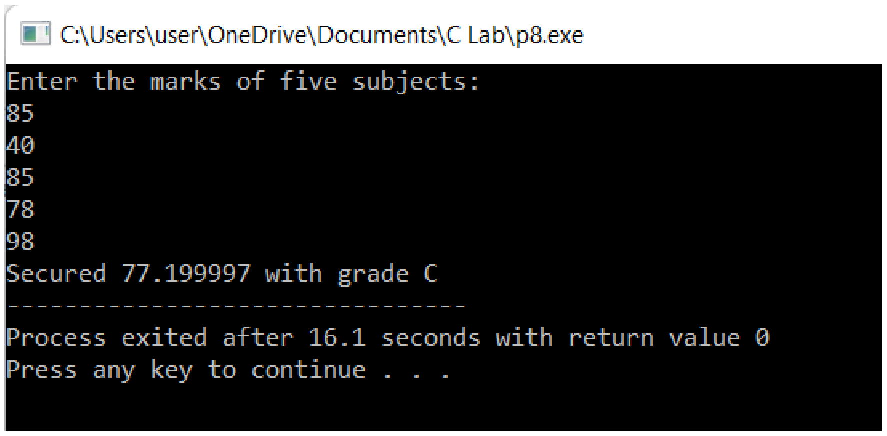


Fig. 8.1 Flowchart to print the grades of students based on their percentage.

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p8.exe
Enter the marks of five subjects:
85
40
85
78
98
Secured 77.199997 with grade C
--------------------------------
Process exited after 16.1 seconds with return value 0
Press any key to continue . . .
```

Fig. 8.2 Input/ Output screenshot of displaying the percentage and grade of a student.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

8.1 What is the conditional statement? How it works?

8.2 Explain nested if else statement.

Experiment no. 9

Objective/Aim: WAP that takes two operands and one operator from the user and perform the operation and prints the result by using Switch statement.

Learning Outcomes (LOs):

LO: Understand the concept of control statements in C.

LO2: Implement switch-case for decision-making.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

- The switch statement allows us to execute one code block among many alternatives.
- In C, jump statements are used to jump from one part of the code to another altering the normal flow of the program. They are used to transfer the program control to somewhere else in the program.

Types of Jump Statements in C

1. break
2. continue
3. goto
4. return

Algorithm:

```
Step 1: Start
Step 2: Read num1, num2, op
Step 3: Print enter + for addition \n  -for subtraction \n * for multiplication \n / for division \n % for modulo
Step 4: Switch(op)
Case '+':
Print addition of num 1 and num 2
and break
Case '-':
Print subtraction of num1 and num2
and break
Case '*':
Print multiplication of num1 and
break
Case '/':
Print division of num 1 and num 2
and break
Case '%':
Print remainder when num 1
```

divide by num2 and break
Default:
Print you made a wrong choice and break
Step 5: Stop

Flowchart:

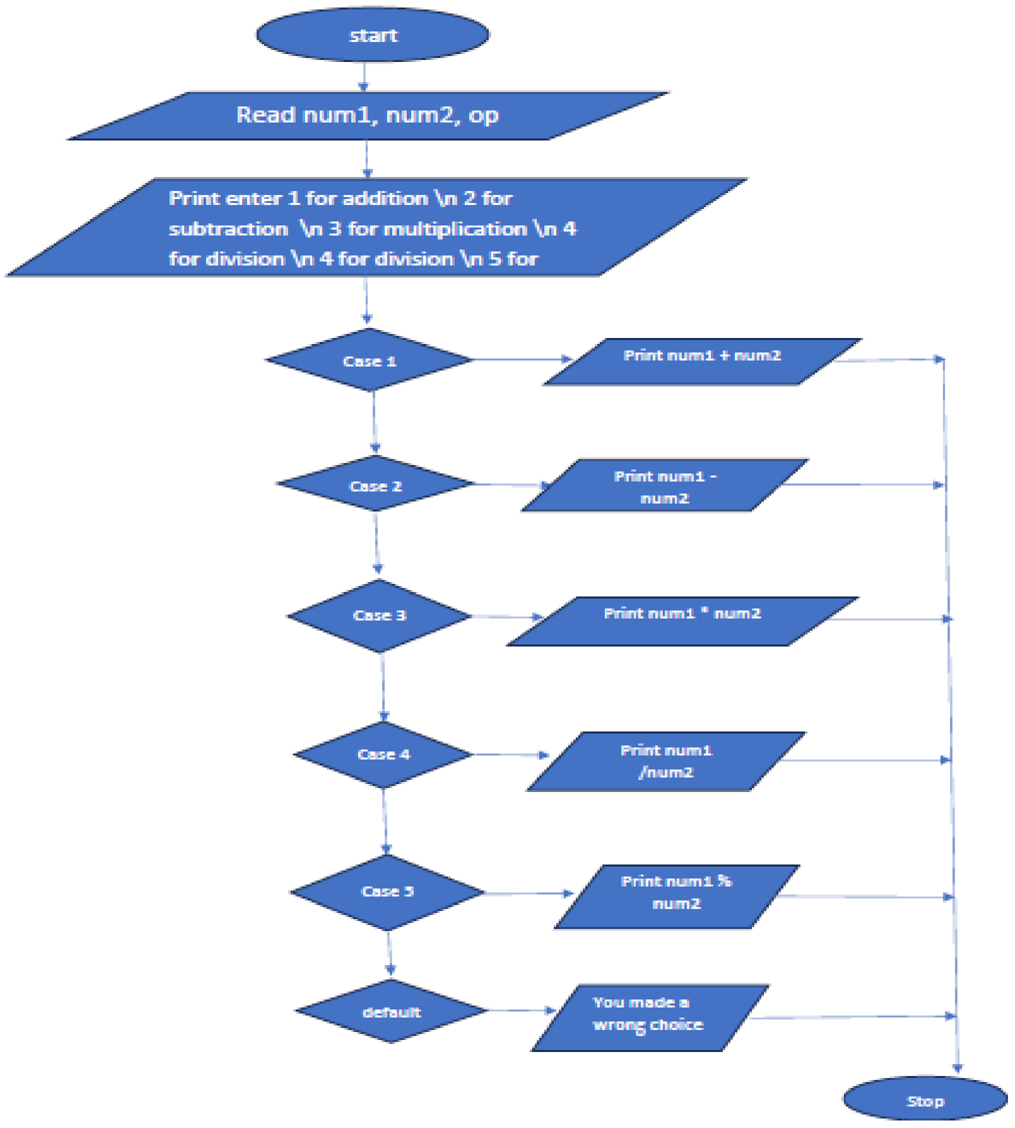
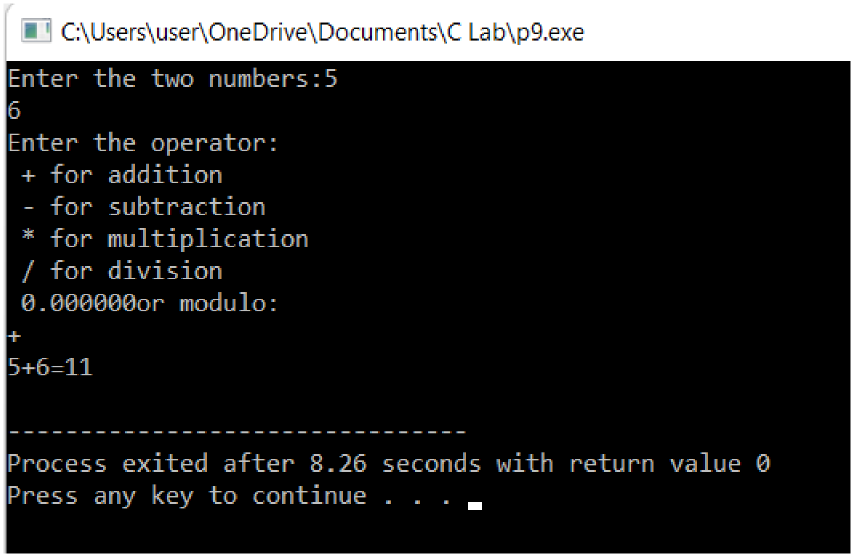


Fig. 9.1 Flowchart to make a small calculator using a Switch case.

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p9.exe
Enter the two numbers:5
6
Enter the operator:
 + for addition
 - for subtraction
 * for multiplication
 / for division
 0.000000or modulo:
+
5+6=11

--------------------------------
Process exited after 8.26 seconds with return value 0
Press any key to continue . . .
```

Fig. 9.2 Input/ Output screenshot of displaying a calculator using Switch case.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

9.1 What is the Jump control statement? How switch case works?

9.2 What is the use of break, continue statement in a program?

9.3 What is the default case in switch statements?

Experiment no. 10

Objective/Aim: WAP to print the sum of all numbers up to a given number.

Learning Outcomes (LOs):

LO1: Understand iterative loops to perform repetitive tasks.

LO2: Implement summation logic using loops.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory: The while Loop is an entry-controlled loop in C programming language. This loop can be used to iterate a part of code while the given condition remains true.

Syntax: The while loop syntax is as follows:

while (test expression)

{ *// body consisting of multiple statements*}

The sum of all positive number up to a given number require a iteration loop till the number n. Every time a variable i increment, add & assign into the sum variable till the required number n.

Formula: sum=sum+i;

Algorithm:

```
Step 1: Start
Step 2: Read num,i,sum
Step 3: Initialise i=0,sum=0
Step 4: Sum=sum+i and i=i+1
Step 5: Repeat steps 4 until i<=num
Step 6: Print sum of numbers
Step 7: Stop
```

Flowchart:

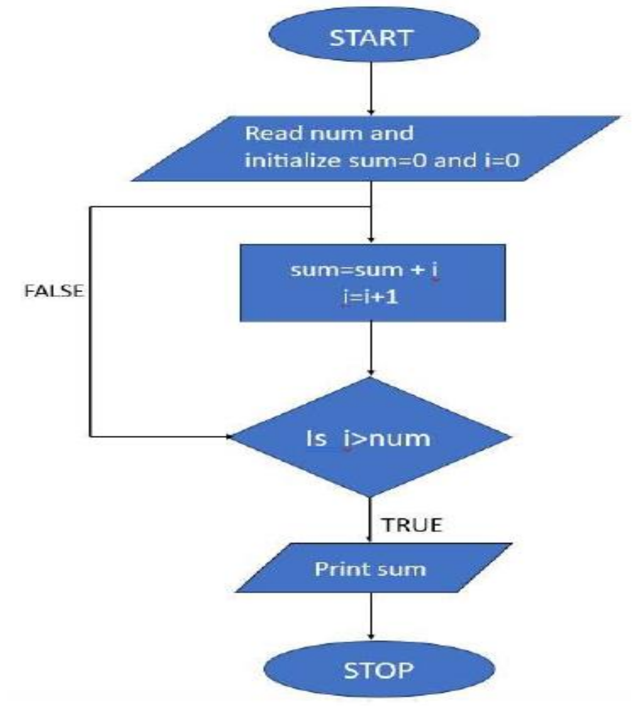
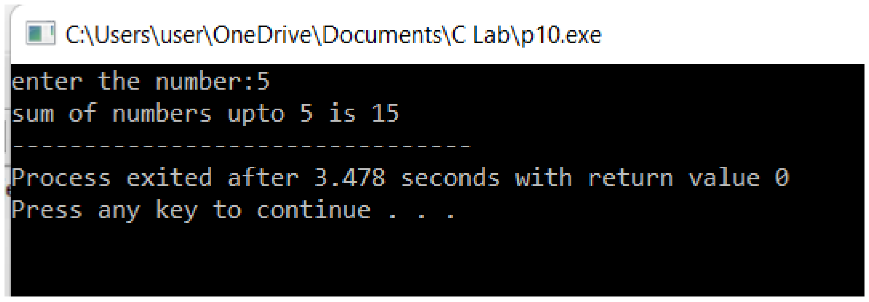


Fig. 10.1 Flowchart to print the sum of all numbers up to a given number.

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p10.exe
enter the number:5
sum of numbers upto 5 is 15
--------------------------------
Process exited after 3.478 seconds with return value 0
Press any key to continue . . .
```

Fig. 10.2 Input/ Output screenshot of displaying the sum of numbers up to a given term.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

10.1 What do you mean by iteration control statements?

10.2 What is while & for loop? How both loops work?

10.3 What is the use of break, continue statement in a program?

10.4 What is the difference between while & do-while loop?

Experiment No. 11

Objective / Aim: WAP to find the factorial of a given number.

Learning Outcomes (LOs):

LO1: Understand the concept of factorials and their mathematical properties.

LO2: Implement iterative or recursive logic for factorial calculation.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

Loops in programming are used to repeat a block of code until the specified condition is met. A loop statement allows programmers to execute a statement or group of statements multiple times without repetition of code

Factorial is a product of all positive integers less than or equal to a given number n. Factorial is denoted by a symbol (!). In simple language, factorial is multiplying the whole numbers until 1.
This program
for example,
Factorial of 5! = 5x4x3x2x1 = 120
In this way, we can calculate the factorial of any positive integer number.

Algorithm:

```
Step 1: Start
Step 2: Read the input number from the user
Step 2: Declare and initialize variables fact = 1 and i=  1
Step 4:  Repeat the loop until  i<=num
fact = fact * i
i=  i++
Step 5:  Print fact to get the factorial of a given number
Step 6: Stop
```

Flowchart:

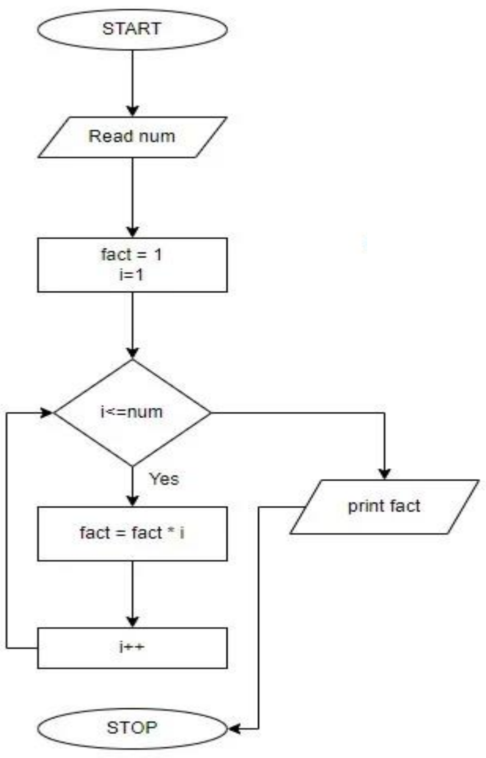
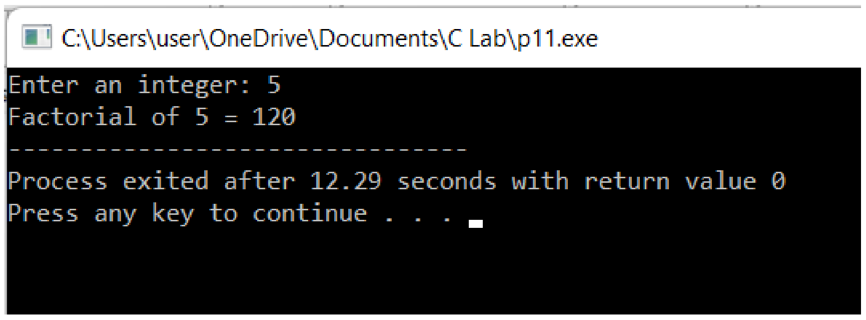


Fig. 11.2 Flowchart to find the factorial of a number.

Input/Output(Screenshot):



```
Enter an integer: 5
Factorial of 5 = 120
--------------------------------
Process exited after 12.29 seconds with return value 0
Press any key to continue . . .
```

Fig. 11.2 Input/ Output screenshot of displaying the Factorial of number.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

11.1 What is the iterative approach of finding a factorial of a number?

11.2 How to count number of zeroes in factorial?

11.3 What are the real-world applications of factorial?

Experiment No. 12

Objective / Aim: WAP to print sum of even and odd numbers from 1 to N numbers.

Learning Outcomes (LOs):

LO1: Apply loop control structures for number series.

LO2: Implement conditional statements for the classification of numbers.

LO3: Enhance logical reasoning in numeric computations.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

Loops in programming are used to repeat a block of code until the specified condition is met. A loop statement allows programmers to execute a statement or group of statements multiple times without repetition of code

The **conditional statements** (also known as decision control structures) such as if, if else, switch, etc. are used for decision-making purposes in C programs.
They are also known as Decision-Making Statements and are used to evaluate one or more conditions and make the decision whether to execute a set of statements or not. These decision-making statements in programming languages decide the direction of the flow of program execution.

Algorithm:

```
Step 1: Start
Step 2: Input the number N
Step 3: Initialize two variables:

    • sum_even = 0 (to store the sum of even numbers)
    • sum_odd = 0 (to store the sum of odd numbers)

Step 4: Loop through numbers from 1 to N:

    • For each number i:
        o If i is even (i.e., $i \bmod 2=0$), then add i to sum_even
        o Else, add i to sum_odd

Step 5: Output sum_even and sum_odd
Step 6: End
```

Flowchart :

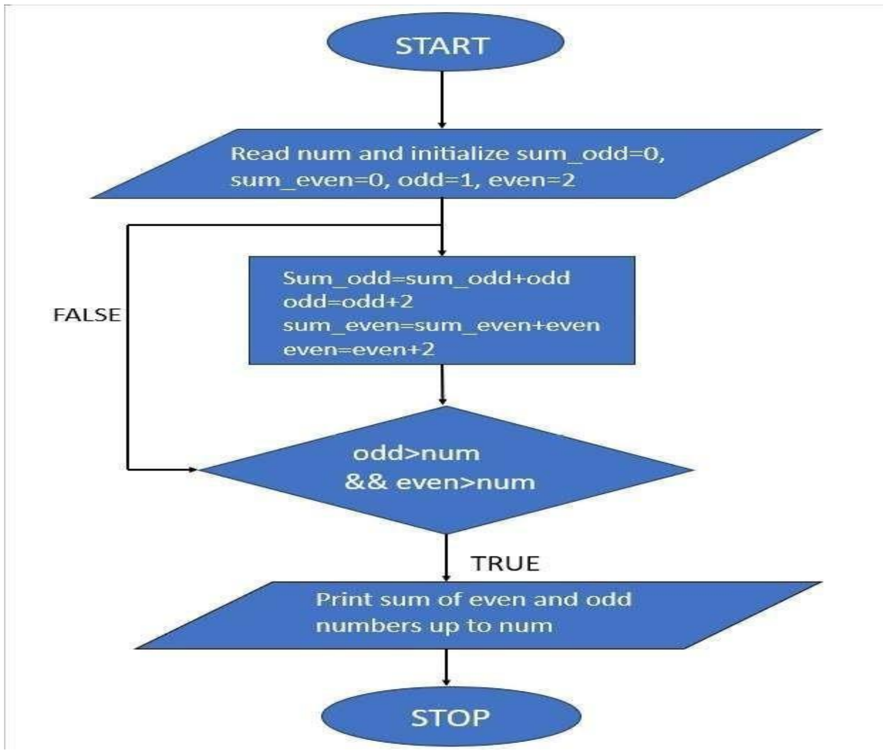
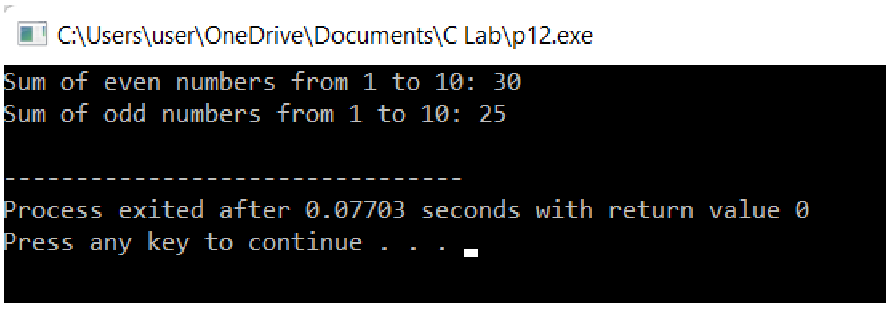

START
Read num and initialize sum_odd=0, sum_even=0, odd=1, even=2
Sum_odd=sum_odd+odd
odd=odd+2
sum_even=sum_even+even
even=even+2
FALSE
odd>num && even>num
TRUE
Print sum of even and odd numbers up to num
STOP

Fig. 12.1 Flowchart to print sum of even and odd numbers.

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p12.exe
Sum of even numbers from 1 to 10: 30
Sum of odd numbers from 1 to 10: 25

--------------------------------
Process exited after 0.07703 seconds with return value 0
Press any key to continue . . .
```

Fig. 12.2 Input/ Output screenshot of displaying the sum of Odd and Even numbers up to a given term.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

12.1 What is an even number, and what is an odd number?

12.2 How do you determine if a number is even or odd in programming?

12.3 What is the modulus operator (%) used for in your algorithm?

12.4 Why do we initialize sum_even and sum_odd to 0 before starting the loop?

Experiment No. 13

Objective / Aim: WAP to print the Fibonacci series.

Learning Outcomes (LOs):

LO1: Understand the concept of sequences and series in programming.

LO2: Implement iterative or recursive solutions for generating Fibonacci numbers.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:
Loops in programming are used to repeat a block of code until the specified condition is met. A loop statement allows programmers to execute a statement or group of statements multiple times without repetition of code
Initial Fibonaccinumbers are 0 and 1.Nextnumbercan be generated by adding twonumbers. So 0+1=1. Therefore next number can be generated by adding two previous. so Fibonacci seriesis011235……

Algorithm:

```
Step 1 : Start
Step 2 : Read n
Step 3 : Initialize f0 = 0, f1 =1, f -0
Step 4 :i=0
Step 5 : while(i<=n) do as follows
printf("%d\t",f0);
            f=f0+f1;
            f0=f1;
            f1=f;
i=i+1;
            If not goto Step 6
Step 6: Stop
```

Flowchart:

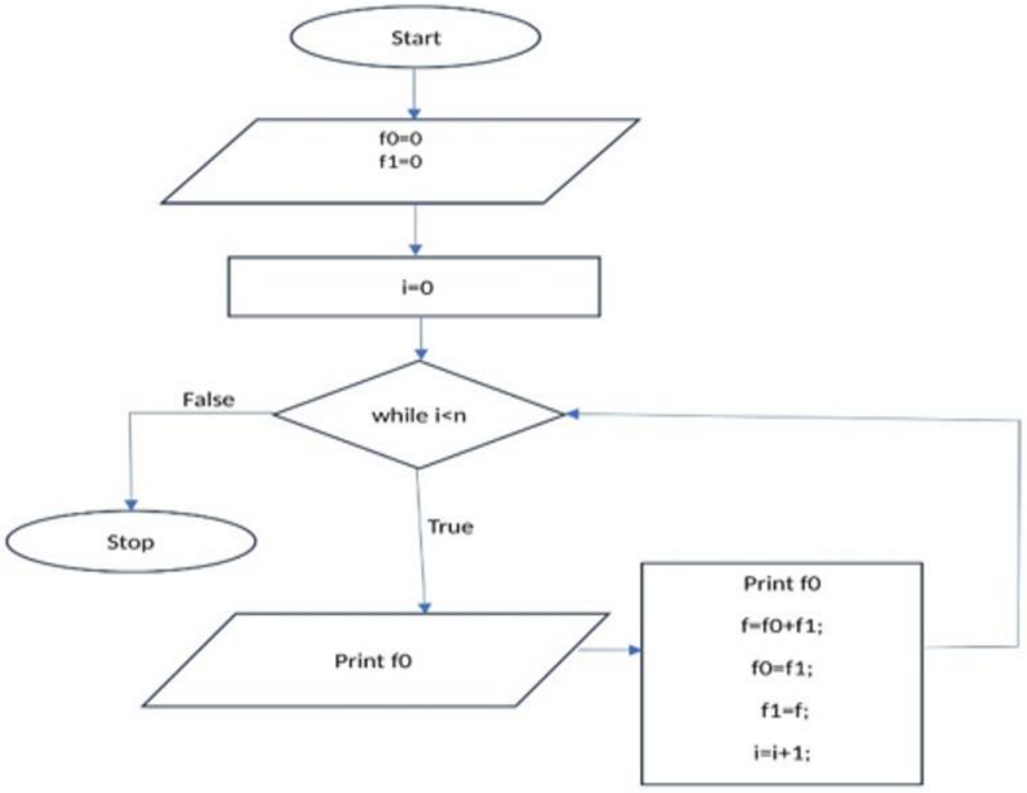
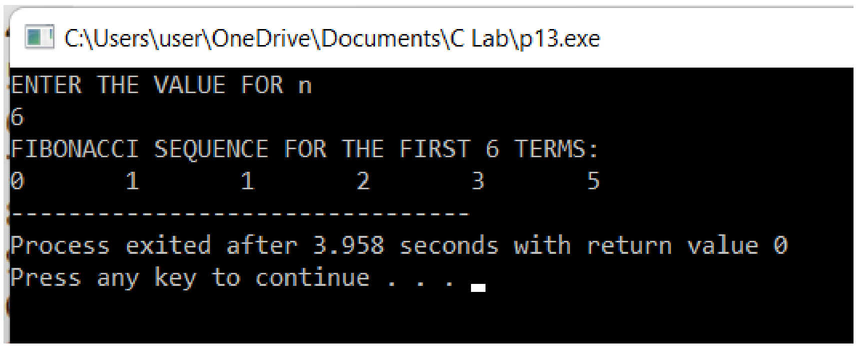


Fig. 13.1 Flowchart to print the Fibonacci Series.

Input/Output(Screenshot):



C:\Users\user\OneDrive\Documents\C Lab\p13.exe

```
ENTER THE VALUE FOR n
6
FIBONACCI SEQUENCE FOR THE FIRST 6 TERMS:
0       1       1       2       3       5
--------------------------------
Process exited after 3.958 seconds with return value 0
Press any key to continue . . .
```

Fig. 13.2 Input/ Output screenshot of displaying the Fibonacci Series up to given terms.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

13.1 What is the Fibonacci series?

13.2 How does the Fibonacci sequence start?

13.3 What is the formula for finding the nth Fibonacci number?

13.4 How does your program calculate the Fibonacci series?

Experiment No. 14

Objective / Aim: WAP to check whether the entered number is prime or not.

Learning Outcomes (LOs):

LO1: Understand the mathematical concept of prime numbers.

LO2: Implement efficient algorithms for checking primality.

LO3: Enhance logical thinking using loops and conditions.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

Loops in programming are used to repeat a block of code until the specified condition is met. A loop statement allows programmers to execute a statement or group of statements multiple times without repetition of code

A prime number is a natural number greater than 1 and is completely divisible only by 1 and itself. In this article, we will learn how to check whether the given number is a prime number or not in C.

Examples
Input: n = 29
Output: 29 is prime
Explanation: 29 has no divisors other than 1 and 29 itself. Hence, it is a prime number.
Input: n = 15
Output: 15 is NOT prime
Explanation: 15 has divisors other than 1 and 15 (i.e., 3 and 5). Hence, it is not a prime number.

Algorithm:

Step 1: Take num as input.

Step 2: Initialize a variable temp to 0.

Step 3: Iterate a “for” loop from 2 to num/2.

Step 4: If num is divisible by loop iterator, then increment temp.

Step 5: If the temp is equal to 0,

Return “Num IS PRIME”.

Else,

Return “Num IS NOT PRIME”.

Flowchart:

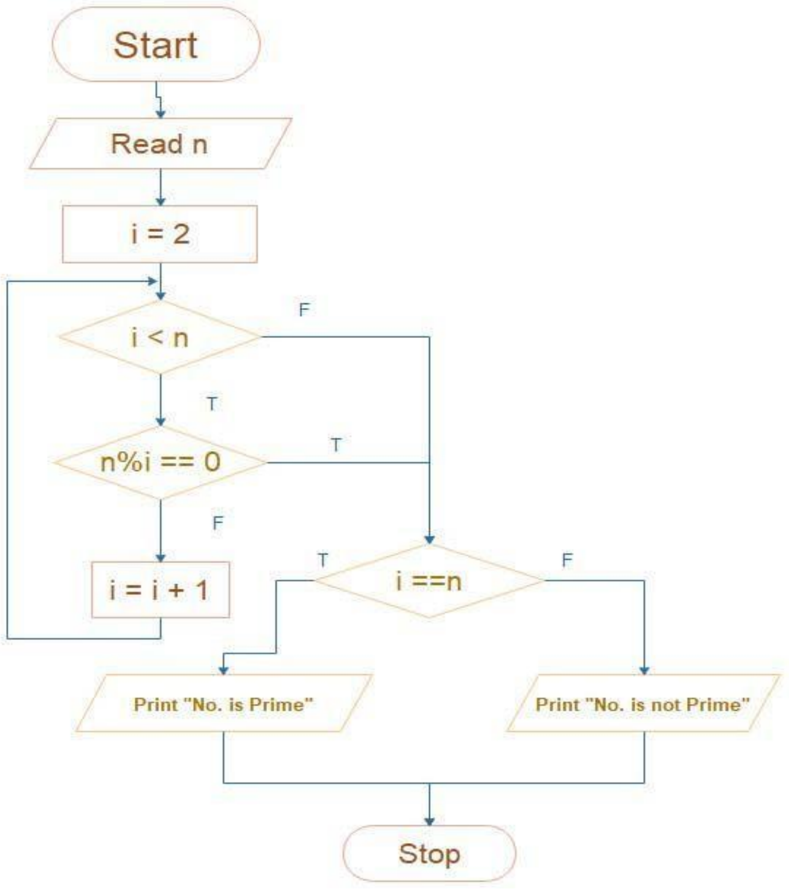
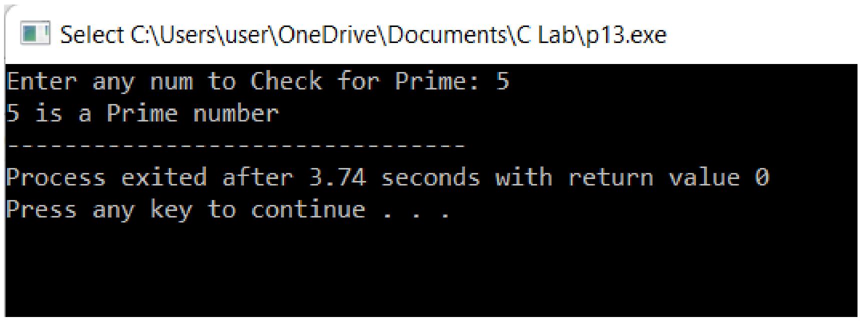


Fig. 14.1 Flowchart to find the Prime number.

Input/Output(Screenshot):



```
Enter any num to Check for Prime: 5
5 is a Prime number
--------------------------------
Process exited after 3.74 seconds with return value 0
Press any key to continue . . .
```

Fig. 14.2 Input/ Output screenshot of displaying the number is Prime or not.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

14.1 What is the iterative approach of finding a factorial of a number?

14.2 How to count number of zeroes in factorial?

14. 2 What are the real world applications of factorial?

Experiment No.15

Objective / Aim: WAP to find the sum of digits of the entered number.

Learning Outcomes (LOs):

LO1: Understand number manipulation techniques in C.

LO2: Apply modular arithmetic to extract digits.

LO3: Improve problem-solving skills for mathematical logic implementation

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

Loops in programming are used to repeat a block of code until the specified condition is met. A loop statement allows programmers to execute a statement or group of statements multiple times without repetition of code. In this C program, we will code Sum of Digits of a Number in C we will allow the user to enter any number and then we will divide the number into individual digits and add those individuals (sum=sum+digit) digits using While Loop.Ex:- number is 231456
2 + 3 + 1 + 4 + 5 + 6 = 21, Sum of digit of a given number is 21

Algorithm:

Step 1: Initialize sum to 0.

Step 2: In a loop:(a) Extract the last digit of number with number % 10 and add it to sum (b) Remove the last digit of number by doing integer division number /= 10.

Step 3: Repeat until number becomes 0.

Step 4: Return sum.

Flowchart:

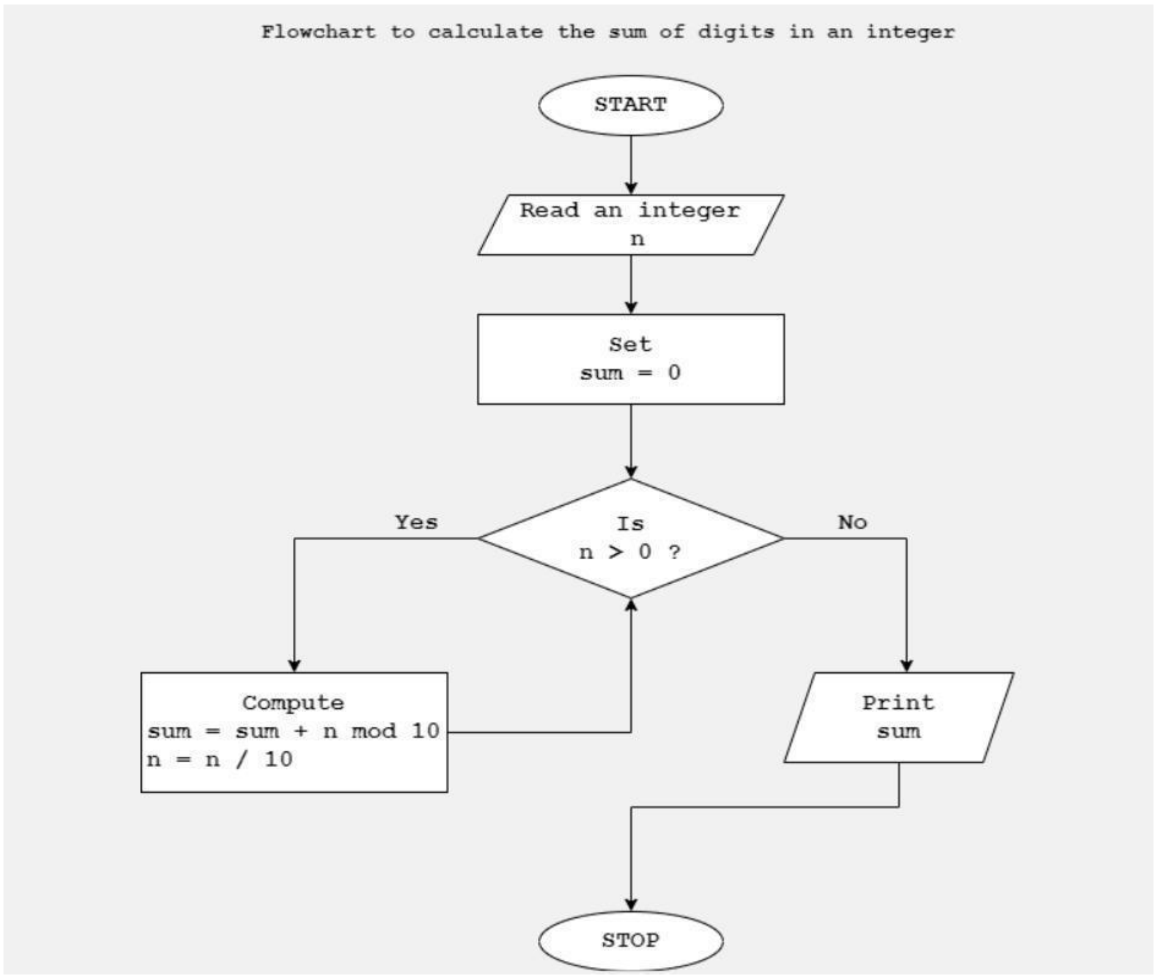
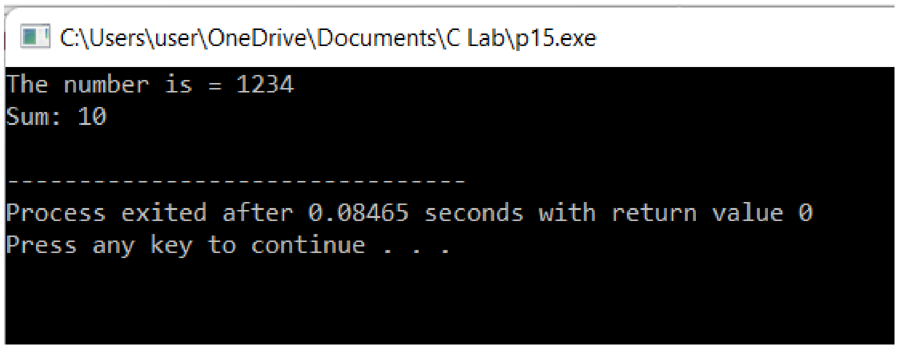


Fig. 15.1 Flowchart to find the sum of digits.

Input/Output(Screenshot):



```
The number is = 1234
Sum: 10

--------------------------------
Process exited after 0.08465 seconds with return value 0
Press any key to continue . . .
```

Fig. 15.2 Input/ Output screenshot of displaying the sum of digits of a number.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

15.1 What operations are mainly used to get the sum of digits?

15.2 How does the program stop when calculating the sum of digits?

15.3 How would you explain the logic to find the sum of digits?

Experiment No.16

Objective / Aim: WAP to find the reverse of a number.

Learning Outcomes (LOs):

LO1: Apply arithmetic operations and loops to reverse a given number.

LO2: Relate number manipulation techniques to real-life scenarios like palindrome checking and data encoding.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

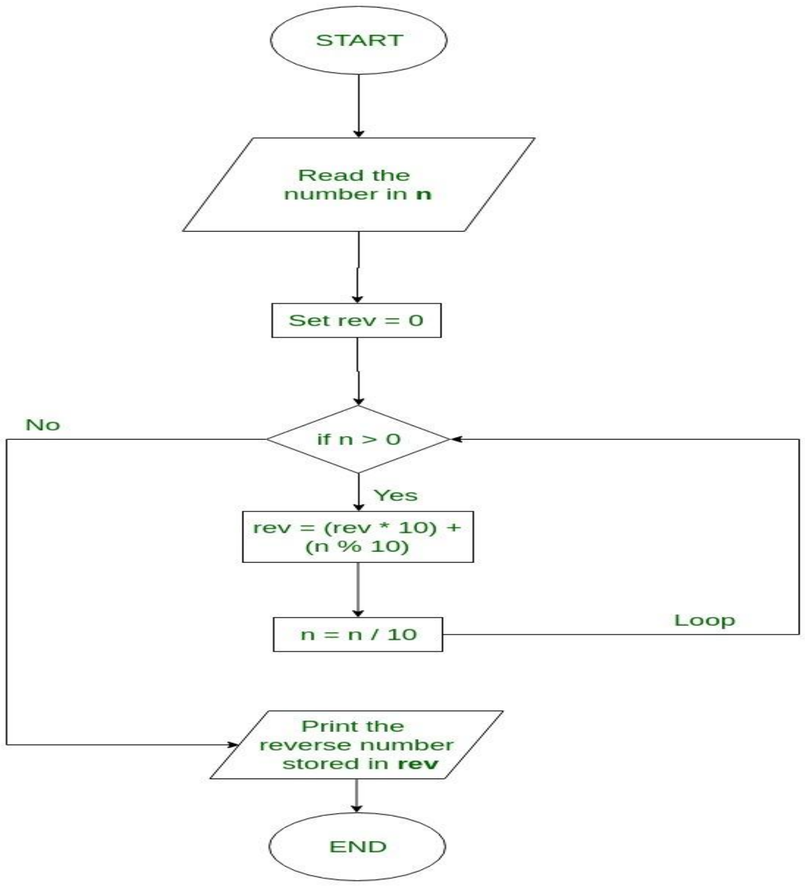
Literature / Theory / Formula:

This program takes integer input from the user. Then the while loop is used until n != 0 is false (0).In each iteration of the loop, the remainder when n is divided by 10 is calculated and the value of n is reduced by 10 times.Inside the loop, the reversed number is computed using: reverse = reverse * 10 + remainder;

Algorithm:

```
Step 1: Input:  num
Step 2: Initialize rev_num = 0
Step 3: Loop while num> 0
    (a) Multiply rev_num by 10 and add remainder of num
        divide by 10 to rev_num
rev_num = rev_num*10 + num%10;
    (b) Divide num by 10
Step 4:  Return rev_num
```

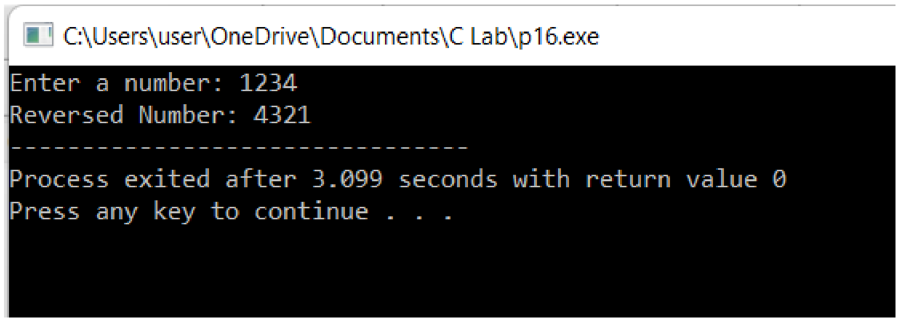
Flowchart:



F

Fig. 16.1 Flowchart to reverse the digits of a number.

Input/Output()Screenshot):


C:\Users\user\OneDrive\Documents\C Lab\p16.exe

```
Enter a number: 1234
Reversed Number: 4321
--------------------------------
Process exited after 3.099 seconds with return value 0
Press any key to continue . . .
```

Fig. 16.2 Input/ Output screenshot of displaying the reverse of a number.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

16.1 What does it mean to reverse a number?

16.2 How would you explain the logic to reverse a number?

16.3. Why do we multiply the reversed number by 10 in each step?

Experiment No. 17

Objective / Aim: Write a program to print Armstrong Number from 1 to 100.

Learning Outcomes (LOs):

LO1: Implement logic to identify Armstrong numbers using digit powers and summation.

LO2: Relate numerical property checking to computational mathematics and pattern recognition.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory: Armstrong numbers are those numbers in which the sum of digits raised to the power of a number of digits in that number will be equal to the number itself.

Note: All one-digit numbers are Armstrong numbers.

Example:

153

$1^3 + 5^3 + 3^3$

$1 + 125 + 27 = 153$

Algorithm:

Step 1) Count the number of digits in the number.
Step 2) Then calculate the sum of digits raised to the power of the number of digits in that number.

Step 3) Check whether sum is equal to the number, if yes then print it.

Step 4) Stop

Flowchart:

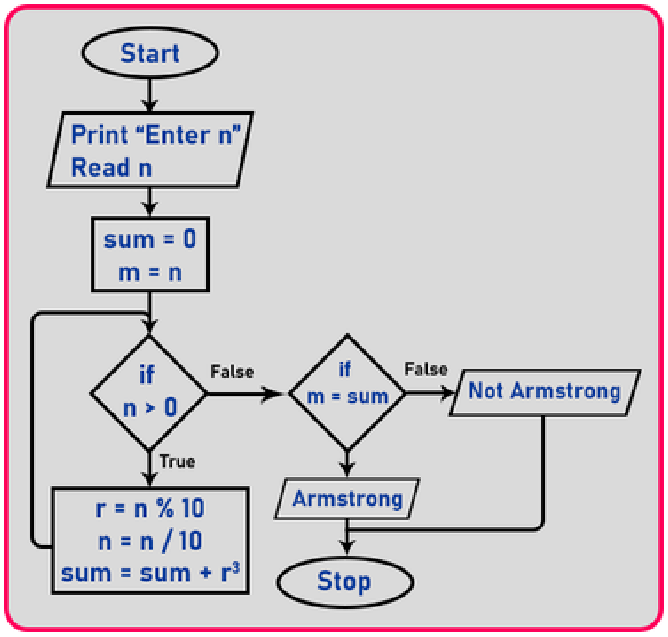
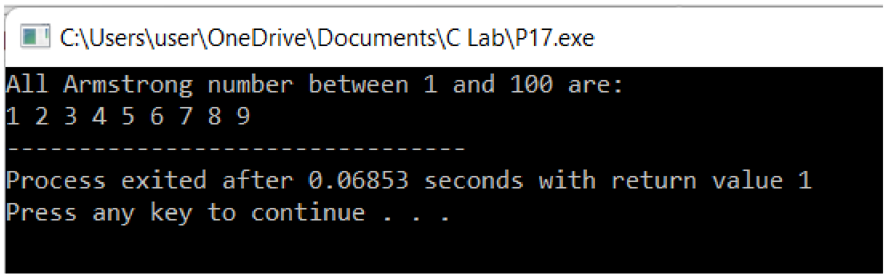


Fig. 17.1 Flowchart to print Armstrong Number.

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\P17.exe
All Armstrong number between 1 and 100 are:
1 2 3 4 5 6 7 8 9
--------------------------------
Process exited after 0.06853 seconds with return value 1
Press any key to continue . . .
```

Fig. 17.2 Input/ Output screenshot of displaying the all Armstrong Number between 1 to 100.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

17.1 What is Armstrong number?

17.2 What is the time complexity to print Armstrong number from 1 to 100?

17.3. What is algorithm and flowchart to print Armstrong number?

17.4 How we can calculate the length of a number?

Experiment no. 18

Objective/Aim: WAP to convert binary number into decimal number and vice versa.

Learning Outcomes (LOs):

LO1: Apply number system conversion algorithms between binary and decimal.

LO2: Relate binary-decimal conversions to real-world applications in computing and digital systems.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

The below diagram explains how to convert (1010) to an equivalent decimal value:

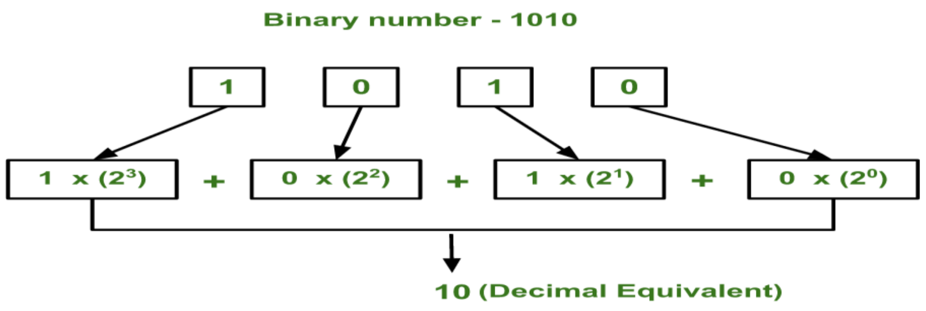


Fig. 18.1 Binary to Decimal conversion.

For Example:

If the decimal number is 10.

Step 1: Remainder when 10 is divided by 2 is zero. Therefore, arr[0] = 0.

Step 2: Divide 10 by 2. New number is 10/2 = 5.

Step 3: Remainder when 5 is divided by 2 is 1. Therefore, arr[1] = 1.

Step 4: Divide 5 by 2. New number is 5/2 = 2.

Step 5: Remainder when 2 is divided by 2 is zero. Therefore, arr[2] = 0.

Step 6: Divide 2 by 2. New number is 2/2 = 1.

Step 7: Remainder when 1 is divided by 2 is 1. Therefore, arr[3] = 1.

Step 8: Divide 1 by 2. New number is 1/2 = 0.

Step 9: Since number becomes = 0. Print the array in reverse order. Therefore the equivalent binary number is 1010.

Algorithm:

```
Step 1: Start
Step 2: read num,n,rem,dec,bin,ch and initialise base =1
Step 3: print enter 1 to perform binary conversion and 2 for decimal conversion
Step 4: if ch=1
            enter the decimal number n=num
Step 5: rem=num%2
            bin=bin+rem*base
            num=num/2
            base=base*10
Step 6: repeat previous step until num>0
            print binary remainder
Step 7: else if ch==2
            enter the binary number n=num
Step 8: rem=num%10
            dec=dec+rem*base
            num=num/10
            base=base*2
Step 9: repeat previous step until num>0
            print the decimal number
Step 10: stop
```

Flowchart:

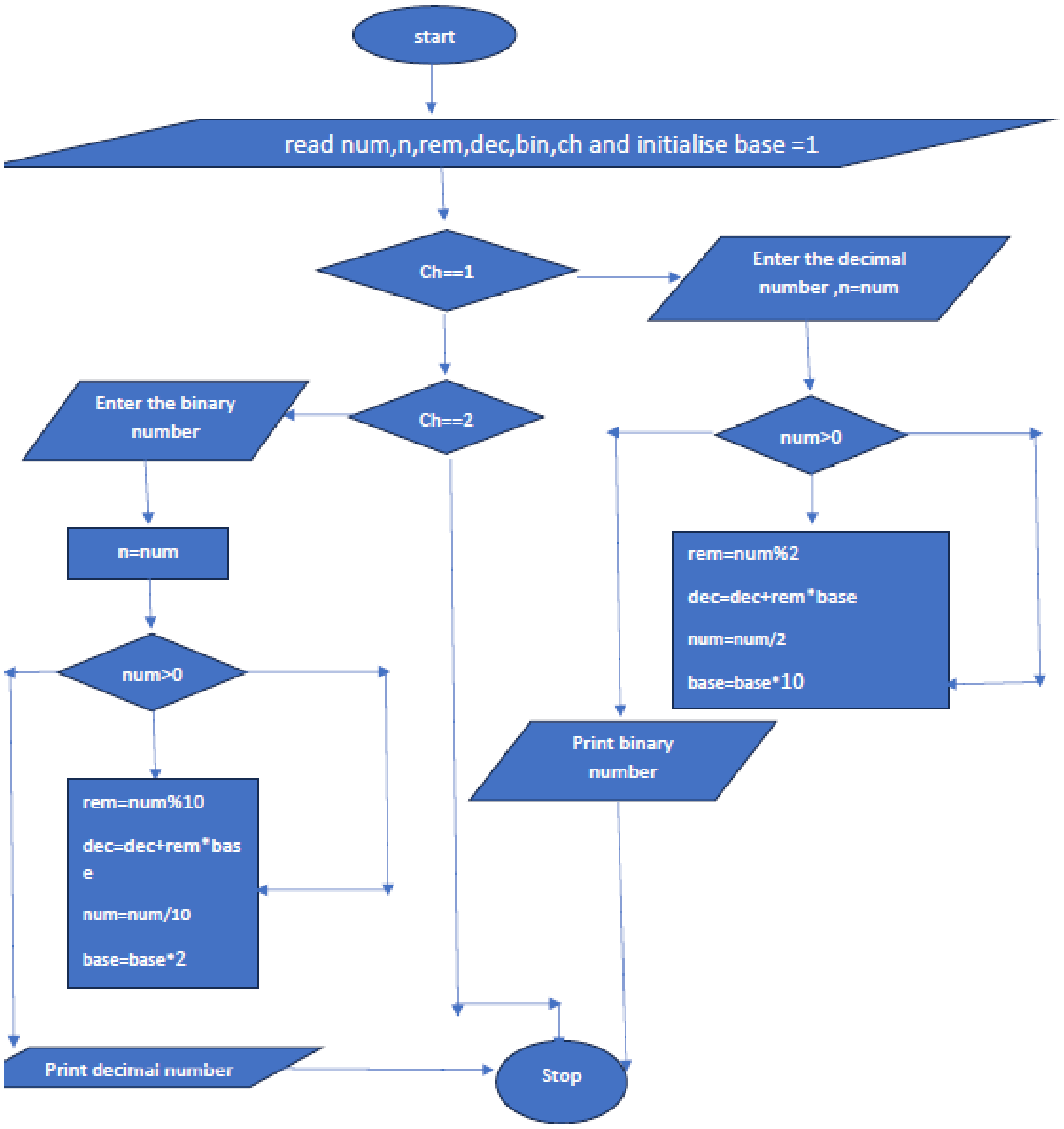
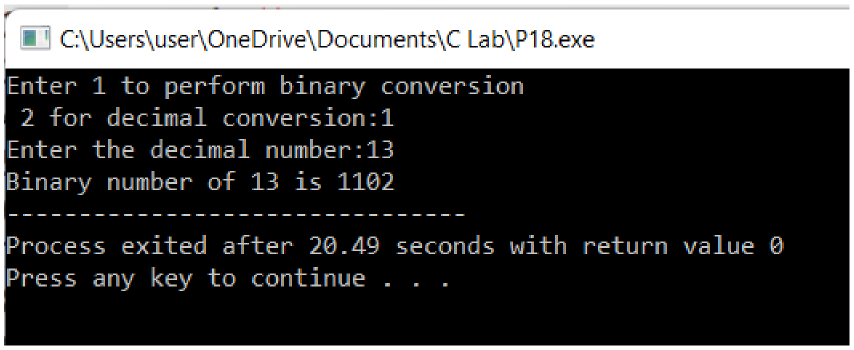


Fig. 18.2 Flow chart for conversion of binary number to decimal and vice versa.

Input/Output(Screenshot):



C:\Users\user\OneDrive\Documents\C Lab\P18.exe

```
Enter 1 to perform binary conversion
 2 for decimal conversion:1
Enter the decimal number:13
Binary number of 13 is 1102
--------------------------------
Process exited after 20.49 seconds with return value 0
Press any key to continue . . .
```

Fig. 18.3 Input/ Output screenshot of displaying the conversion of Binary to Decimal and vice-versa.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

18.1 What are the binary number & decimal number?

18.2 What do you mean by binary conversion?
18.3 What is the process to convert binary to decimal or vice-versa?

18.4 What are the different operators used in this program?

Experiment no. 19

Objective/Aim: WAP that simply takes elements of the array from the user and finds the sum of these elements.

Learning Outcomes (LOs):

LO1: Demonstrate the ability to declare, initialize, and access array elements.

LO2: Integrate array traversal and summation logic to solve real-life data aggregation problems.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature / Theory / Formula:

- Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.
- To create an array, define the data type (like int) and specify the name of the array followed by **square brackets []**.
- To insert values to it, use a comma-separated list inside curly braces, and make sure all values are of the same data type: Ex: int Arr[] = {25, 50, 75, 100};

  Access the Elements of an Array

  - To access an array element, refer to its **index number**.
  - Array indexes start with **0**: [0] is the first element. [1] is the second element, etc.
  - This statement accesses the value of the **first element [0]** in Arr:
    Example:

    ```
    int Arr[] = {25, 50, 75, 100};
    printf("%d", Arr[0]); // Output 25
    ```

  - To calculate sum of all the array elements, we will use, sum=sum+ Arr[i];

Algorithm:

```
Step 1: Start
Step 2: read Arr[50], i, size and initialize sum=0
Step 3: initialize i=0
Step 4: enter the element at arr[i], print the element, sum=sum+Arr[i]
Step 5: increment i by 1
```

Step 6: repeat step 4 and 5 until i<size

Step 7: print sum

Step 8: Stop

Flowchart:

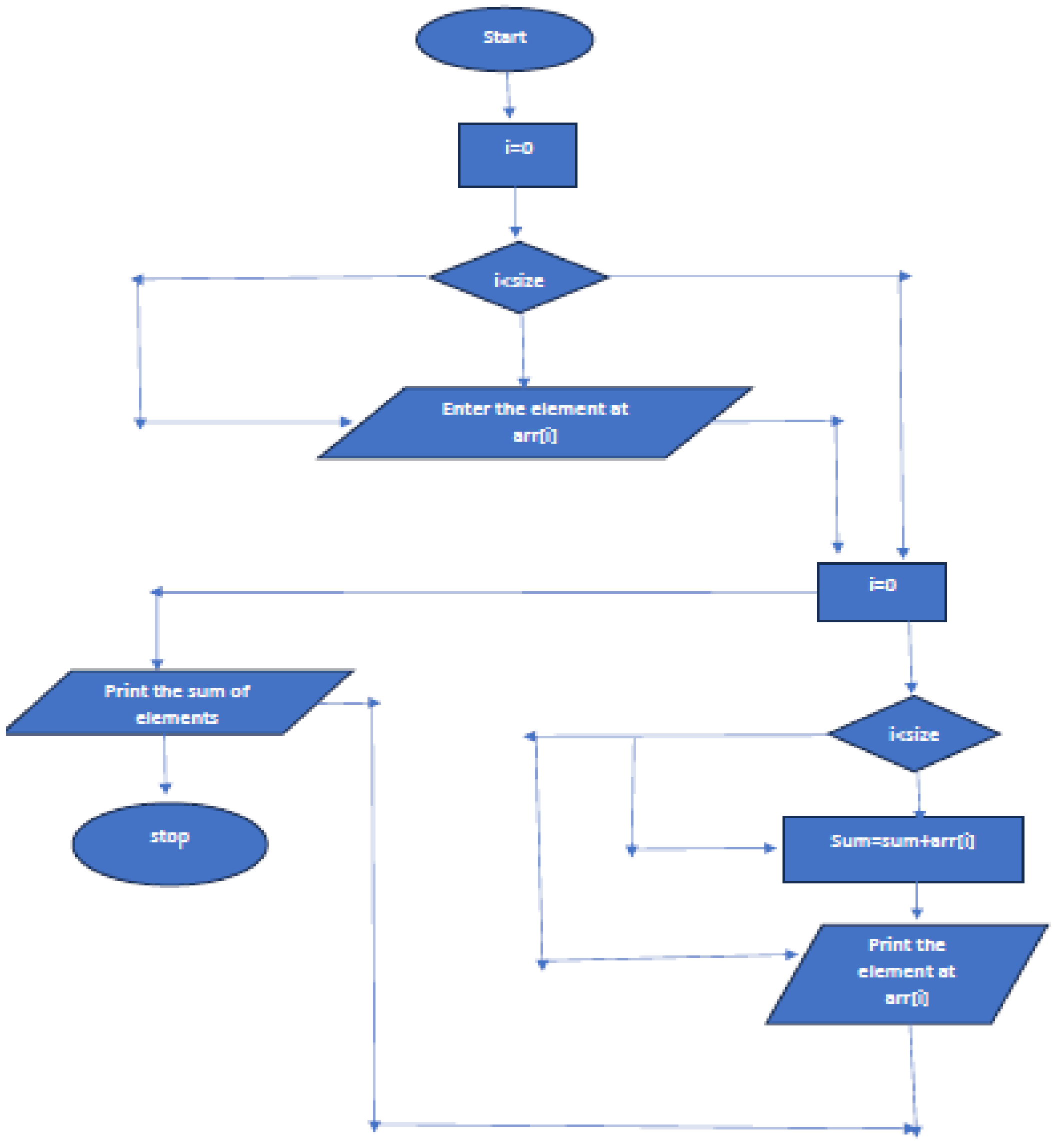
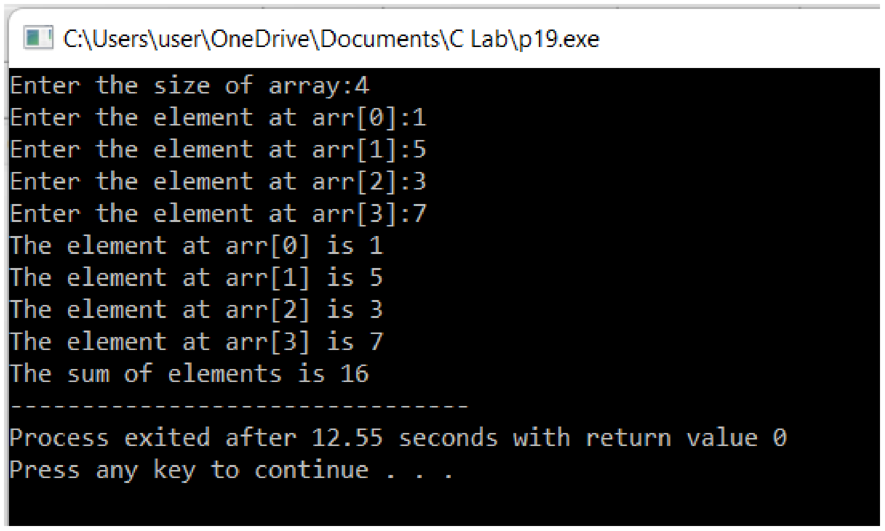

Start
i=0
i<size
Enter the element at arr[i]
i=0
Print the sum of elements
stop
i<size
Sum=sum+arr[i]
Print the element at arr[i]

Fig. 19.1 Flowchart for sum of elements of an array.

Input/Output(Screenshot):



C:\Users\user\OneDrive\Documents\C Lab\p19.exe

```
Enter the size of array:4
Enter the element at arr[0]:1
Enter the element at arr[1]:5
Enter the element at arr[2]:3
Enter the element at arr[3]:7
The element at arr[0] is 1
The element at arr[1] is 5
The element at arr[2] is 3
The element at arr[3] is 7
The sum of elements is 16
--------------------------------
Process exited after 12.55 seconds with return value 0
Press any key to continue . . .
```

Fig. 19.2 Input/ Output screenshot of displaying sum of elements of an array.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

19.1 What are the Array Data Structure?

19.2 What do you mean by index in array?

19.3 What is while & for loop? How both loops work?

19.4 What is the difference between while & for loop?

Experiment No. 20

Objective / Aim: Write a program that inputs two arrays and saves sum of corresponding elements of these arrays in a third array and print them.

Learning Outcomes (LOs):

LO1: Apply array concepts to store, access, and manipulate multiple datasets simultaneously.

LO2: Relate array operations to practical scenarios where merging or combining multiple data sources is required.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory: An array in C is the collection of similar data items together in a contiguous memory location. You can locate each element with the help of its index. In the array, the index starts at 0 for the first elements and then progresses one by one.

As per the problem statement we have to find the sum of two different arrays with respect to index and store it into a third array. Suppose a1[] is first array, a2[] is second array and a3[] is third array, then sum of a1[] and a2[] should be stored in a3[] i.e.

a1[0] + a2[0] = a3[0]

a1[1] + a2[1] = a3[1]

a1[2] + a2[2] = a3[2] and so on.

Let's start!

Note − Two array lengths must be the same and array elements should be numeric.

Suppose a1[] array is {5, 6, 3, 2, 4, 11} and a2[] array is {3, 9, 5, 21, 19, 2}

After adding the two arrays, the result will be −

Resultant array is: [8, 15, 8, 23, 23, 13]

Algorithm:

Step 1) Declare the three matrix say M1,M2, M3.

Step 2) Enter the elements of Matrix M1 & M2.

Step 3) Initiate the loop variable say i=0.

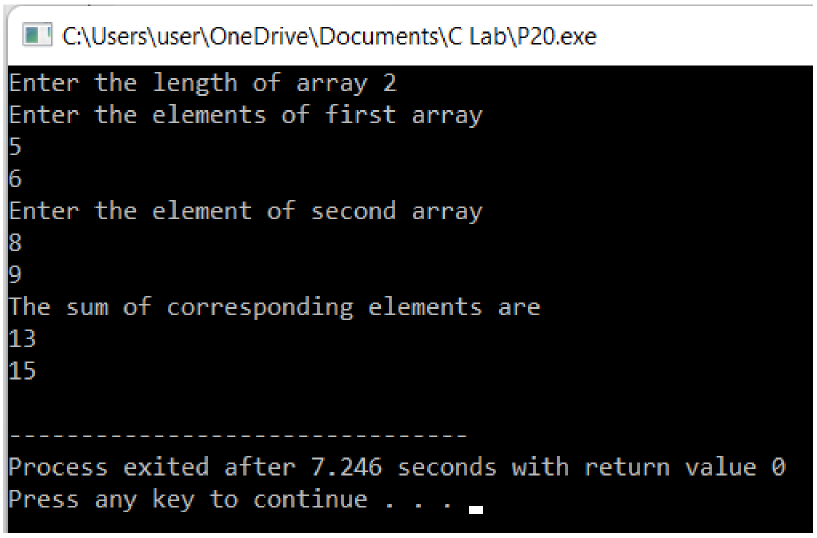
Step 4) If 'i' less than size of array, go to step 5. Else, go to step 6.

Step 5) Calculate the sum of corresponding elements of M1 & M2, and store it to Corresponding index of M3 (M3[i]=M1[i]+M2[i])

Step 6) Display the elements of resultant matrix i.e M3.

Step 7) Stop

Input/Output(Screenshot):



C:\Users\user\OneDrive\Documents\C Lab\P20.exe

```
Enter the length of array 2
Enter the elements of first array
5
6
Enter the element of second array
8
9
The sum of corresponding elements are
13
15

--------------------------------
Process exited after 7.246 seconds with return value 0
Press any key to continue . . .
```

Fig. 20.1 Input/ Output screenshot of displaying sum of elements of two array.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

20.1 What is array?

20.2 What are the applications of array?

20.3 What are the types of array?

20.4 What is the syntax to declare and initialize the array?

20.5 What are the pros and cons of the array?

Experiment No. 21

Objective / Aim: WAP to find the minimum and maximum element of the array.

Learning Outcomes:

LO1: Identify the largest or smallest element in an array.

LO2: Implement traversal logic to compare all elements.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory: An array is a collection of elements of the same data type stored in contiguous memory locations. To solve many programming problems efficiently, we often need to determine the smallest (minimum) and largest (maximum) values stored in an array.

The **minimum** is the smallest value among all elements.

The **maximum** is the largest value among all elements.

For example, in the array:
[10, 3, 7, 1, 25]

Minimum = 1

Maximum = 25

Algorithm:

Step1: Start

Step 2: Declare an array arr[] and input its size n.

Step 3: Input n elements into the array.

Step 4: Initialize two variables:

- min = arr[0]
- max = arr[0]

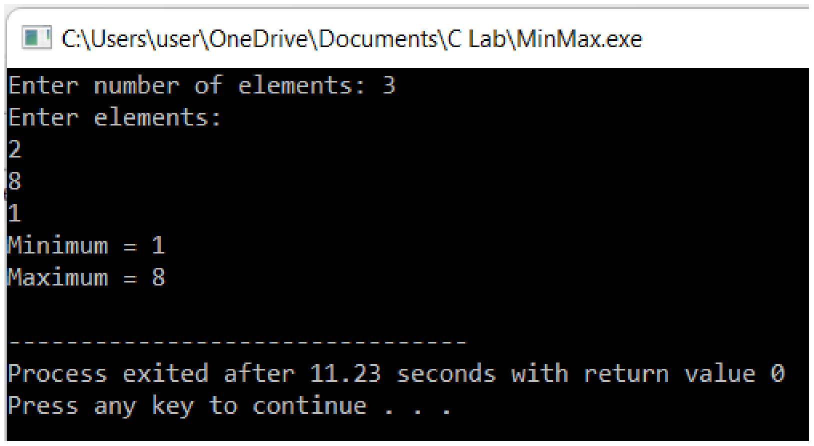
Step 5: Repeat steps 6–8 for i = 1 to n-1:
- If arr[i] < min, then min = arr[i]
- If arr[i] > max, then max = arr[i]

Step 6: End of loop.

Step 7: Print min and max.

Step 8: Stop

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\MinMax.exe
Enter number of elements: 3
Enter elements:
2
8
1
Minimum = 1
Maximum = 8

--------------------------------
Process exited after 11.23 seconds with return value 0
Press any key to continue . . .
```

Fig. 21.1 Input/ Output screenshot of displaying Max and Min element of an array.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

21.1 What is linear search algorithm?

Experiment No. 22

Objective / Aim: Write a program to search an element in an array using Linear Search.

Learning Outcomes:

LO1: Recall the concept of searching in arrays.

LO2: Implement linear search to locate an element.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory: Linear search is a simple algorithm for searching for an item in a data set by examining each element in order until it finds a match. It's a good choice for small arrays or single searches in unsorted arrays.

- **Start:** Begin at the first element of the collection of elements.
- **Compare:** Compare the current element with the desired element.
- **Found:** If the current element is equal to the desired element, return true or index to the current element.
- **Move:** Otherwise, move to the next element in the collection.
- **Repeat:** Repeat steps 2-4 until we have reached the end of collection.
- **Not found:** If the end of the collection is reached without finding the desired element, return that the desired element is not in the array.

Algorithm:

Step 1) Read the search item, "item."

Step 2) Initiate i=0 and index=-1

Step 3) If i<N, go to step 4. Else, go to step 8.

Step 4) If Data[i] equals to "item," then go to step 5. Else go to step 6.

Step 5) Index = i (As the item is found at index no i). Go to step 8.

Step 6) i = i +1.

Step 7) Go to step 3.

Step 8) Stop

Flowchart:

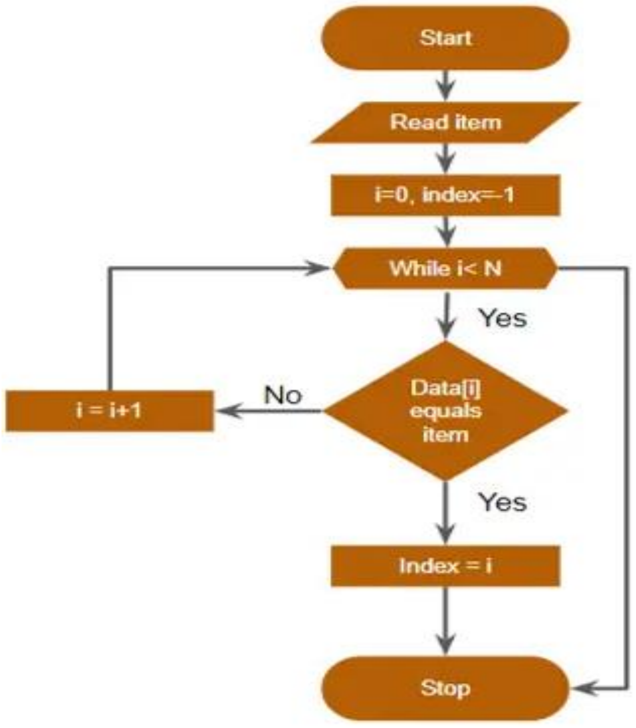
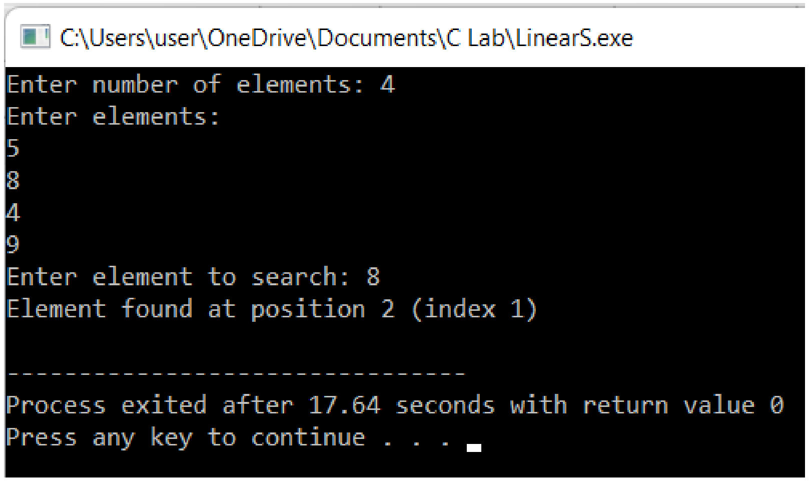


Fig. 22.1 Flowchart of Linear Search.

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\LinearS.exe
Enter number of elements: 4
Enter elements:
5
8
4
9
Enter element to search: 8
Element found at position 2 (index 1)

--------------------------------
Process exited after 17.64 seconds with return value 0
Press any key to continue . . .
```

Fig. 22.2 Input/ Output screenshot of displaying searching of an element using Linear search.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

22.1 How does linear search algorithm work?

22.2 What is the time complexity of linear search algorithm?

22.3 When is linear search algorithm preferred over other searching algorithms?

22.4 What are the advantages of linear search algorithm?

Experiment No. 23

Objective / Aim: Write a program to sort the elements of the array in ascending order using the Bubble Sort technique.

Learning Outcomes:

LO1: Understand the concept of adjacent element comparison.

LO2: Implement bubble sort to arrange elements in ascending order.

Apparatus Used: Turbo c compiler, gcc compiler (IDE- vs. code, dev c)

Theory: Bubble sort is a simple algorithm for sorting a list of elements by repeatedly comparing and swapping adjacent elements until the list is in order.

1. Start at the beginning of the list
2. Compare the first value with the next one
3. If the first value is larger, swap the positions of the two values
4. Repeat until there are no more items to compare
5. Go back to the start of the list

Algorithm:

We assume list is an array of n elements. We further assume that swap function swaps the values of the given array elements.

Step 1 − Check if the first element in the input array is greater than the next element in the array.
Step 2 − If it is greater, swap the two elements; otherwise move the pointer forward in the array.
Step 3 − Repeat Step 2 until we reach the end of the array.
Step 4 − Check if the elements are sorted; if not, repeat the same process (Step 1 to Step 3) from the last element of the array to the first.
Step 5 − The final output achieved is the sorted array.

Flowchart:

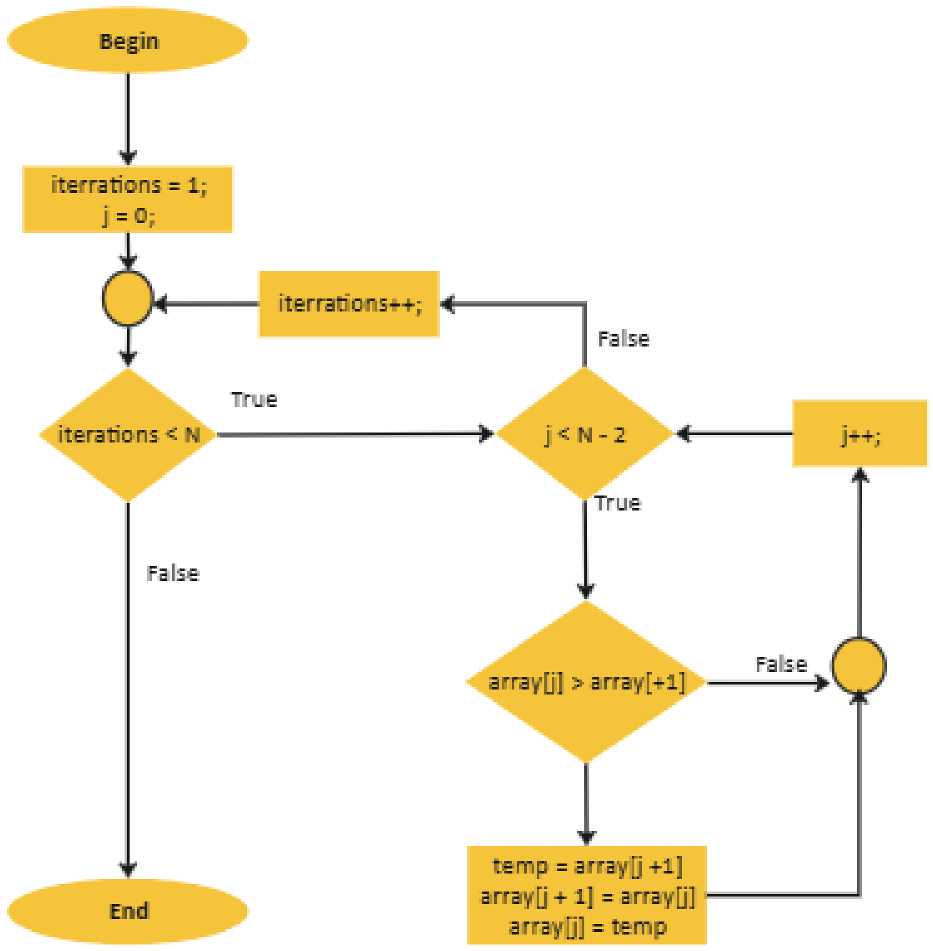
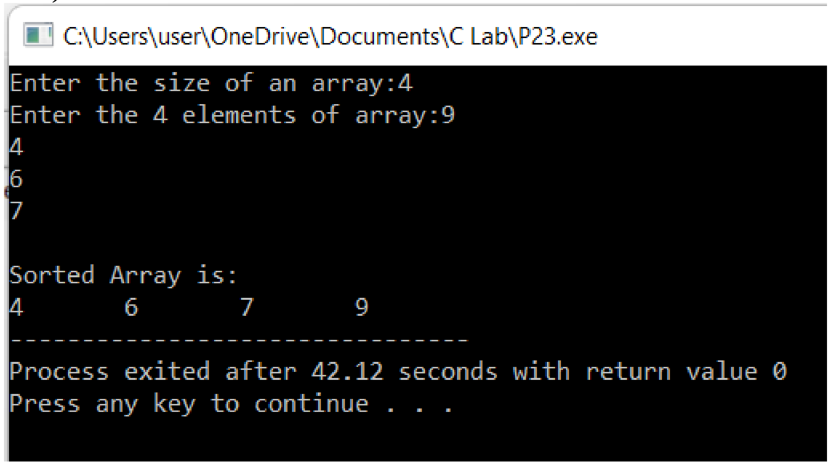


Fig. 23.1 Flowchart of Bubble Sort.

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\P23.exe
Enter the size of an array:4
Enter the 4 elements of array:9
4
6
7

Sorted Array is:
4       6       7       9
--------------------------------
Process exited after 42.12 seconds with return value 0
Press any key to continue . . .
```

Fig. 23.2 Input/ Output screenshot of displaying sorting of an array using Bubble sort.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

23.1 What is bubble sort?

23.2 Is bubble sort stable?

23.3 What is the best-case runtime complexity of bubble sort?

23.4 Why is bubble sort inefficient for large data sets?

Experiment No. 24

Objective / Aim: Write a program to sort the elements of the array in ascending order using the insertion sort and selection sort with function.

Learning Outcomes:

LO1: Implement selection and insertion sort step by step.

LO2: Compare the efficiency of bubble, selection, and insertion sort.

LO3: Decide which sorting algorithm is best suited for different contexts.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory: *Insertion* Sort sequentially inserts elements into a sorted sub array, whereas Selection Sort finds and places the smallest element in each iteration, generally resulting in fewer swaps.

Algorithm (Insertion Sort):

Step 1 - If the element is the first element, assume that it is already sorted. Return 1.
Step2 - Pick the next element, and store it separately in a **key.**
Step3 - Now, compare the **key** with all elements in the sorted array.
Step 4 - If the element in the sorted array is smaller than the current element, then move to the next element. Else, shift greater elements in the array towards the right.
Step 5 - Insert the value.
Step 6 - Repeat until the array is sorted.

Flowchart:

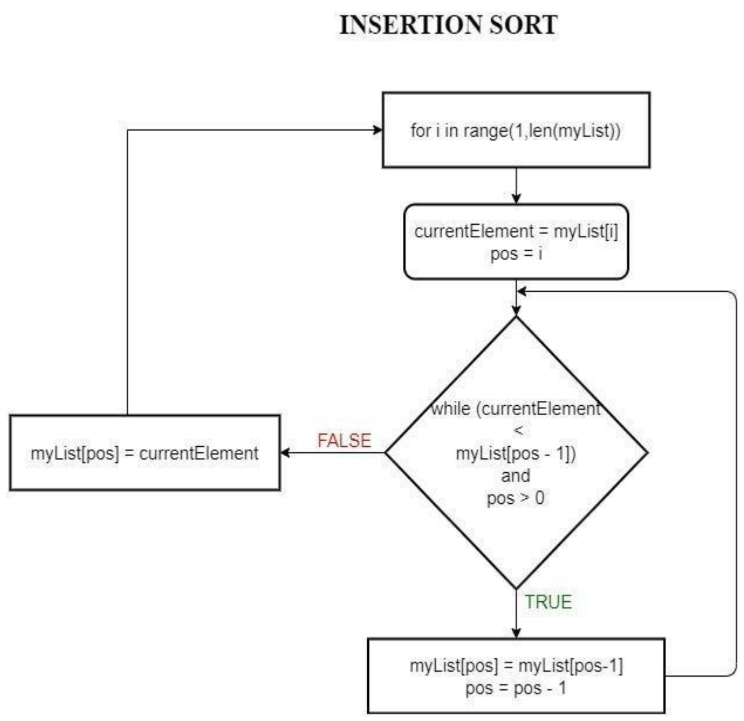
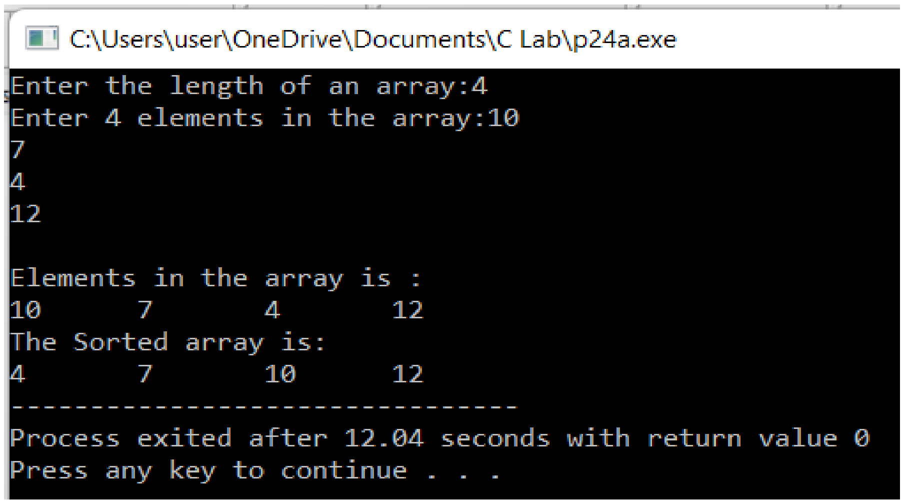

INSERTION SORT
for i in range(1,len(myList))
currentElement = myList[i]
pos = i
while (currentElement < myList[pos - 1]) and pos > 0
FALSE
myList[pos] = currentElement
TRUE
myList[pos] = myList[pos-1]
pos = pos - 1

Fig. 24.1 Flowchart of insertion sort.

Input/Output(Screenshot):



C:\Users\user\OneDrive\Documents\C Lab\p24a.exe

```
Enter the length of an array:4
Enter 4 elements in the array:10
7
4
12

Elements in the array is :
10      7       4       12
The Sorted array is:
4       7       10      12
--------------------------------
Process exited after 12.04 seconds with return value 0
Press any key to continue . . .
```

Fig. 24.2 Input/ Output screenshot of displaying sorting of an array using Insertion sort.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Algorithm (Selection Sort):

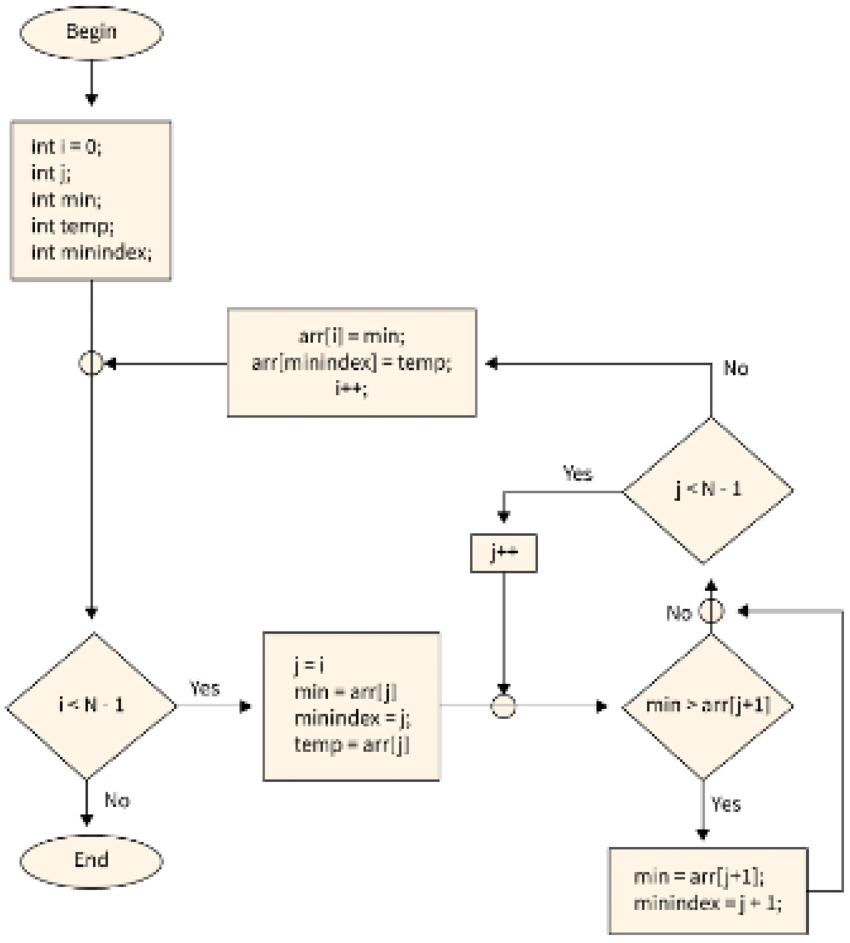
```
SELECTION SORT(arr, n)
Step 1: Repeat Steps 2 and 3 for i = 0 to n-1
Step 2: CALL SMALLEST(arr, i, n, pos)
Step 3: SWAP arr[i] with arr[pos]
[END OF LOOP]
Step 4: EXIT

SMALLEST (arr, i, n, pos)
Step 1: [INITIALIZE] SET SMALL = arr[i]
Step 2: [INITIALIZE] SET pos = i
Step 3: Repeat for j = i+1 to n
if (SMALL > arr[j])
```

```
    SET SMALL = arr[j]
SET pos = j
[END OF if]
[END OF LOOP]
Step 4: RETURN pos
```

Procedure:

1. First we find the smallest element and swap it with the first element. This way we get the smallest element at its correct position.
2. Then we find the smallest among remaining elements (or second smallest) and move it to its correct position by swapping.
3. We keep doing this until we get all elements moved to correct position.

Flowchart:



Input/Output(Screenshot):

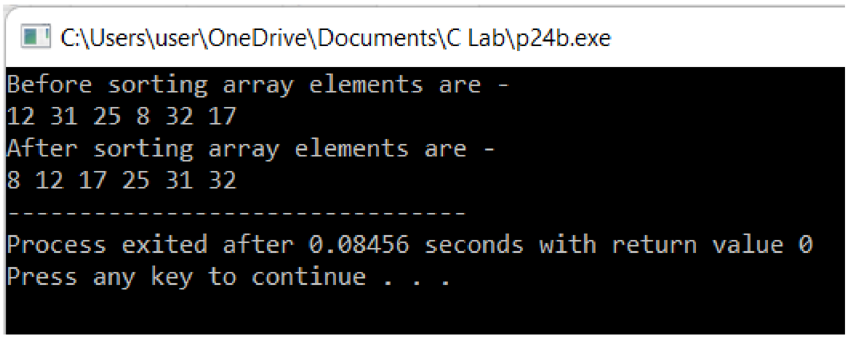

```
C:\Users\user\OneDrive\Documents\C Lab\p24b.exe
Before sorting array elements are -
12 31 25 8 32 17
After sorting array elements are -
8 12 17 25 31 32
--------------------------------
Process exited after 0.08456 seconds with return value 0
Press any key to continue . . .
```

Fig. 24.4 Input/ Output screenshot of displaying sorting of an array using Selection sort.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

24.1 What are the Boundary Cases of the Insertion Sort algorithm?

24.2 What is the Algorithmic Paradigm of the Insertion Sort algorithm?

24.3 Is Insertion Sort a stable algorithm?

24.4 When is the Insertion Sort algorithm used?

Experiment No. 25

Objective / Aim: Write a program to search an element in an array using Binary Search.

Learning Outcomes:

LO1: Understand the requirement of sorted arrays.

LO2: Implement the binary search algorithm stepwise.

LO3: Contrast binary search with linear search and extend to other structures.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory: Binary Search Algorithm is a searching algorithm used in a sorted array by **repeatedly dividing the search interval in half**. The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(log N).

Algorithm:

Binary_Search(a, lower_bound, upper_bound, val) // 'a' is the given array, 'lower_bound' is the index of the first array element, 'upper_bound' is the index of the last array element, 'val' is the value to search

```
Step 1: set beg = lower_bound, end = upper_bound, pos = - 1
Step 2: repeat steps 3 and 4 while beg <=end
Step 3: set mid = (beg + end)/2
Step 4: if a[mid] = val
set pos = mid
print pos
go to step 6
else if a[mid] > val
set end = mid - 1
else
set beg = mid + 1
[end of if]
[end of loop]
Step 5: if pos = -1
print "value is not present in the array"
[end of if]
Step 6: exit
```

Flowchart:

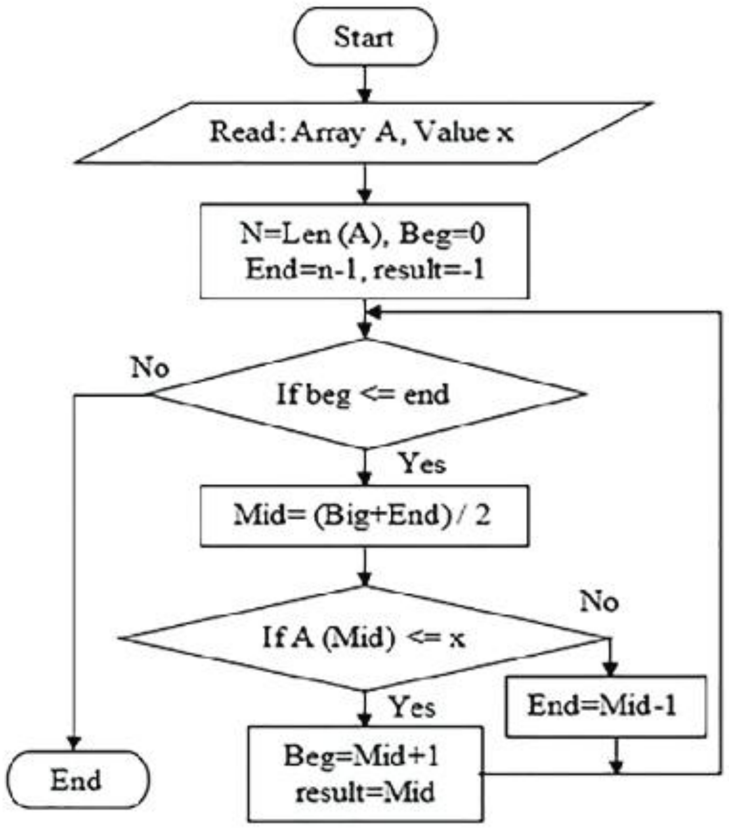
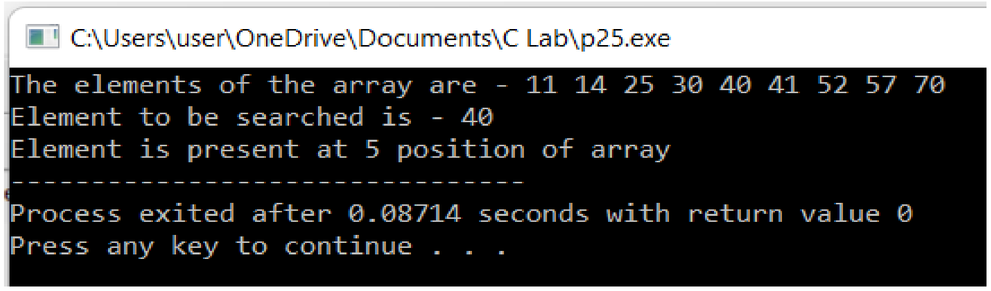


Fig. 25.1 Flowchart of Binary Search.

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p25.exe
The elements of the array are - 11 14 25 30 40 41 52 57 70
Element to be searched is - 40
Element is present at 5 position of array
--------------------------------
Process exited after 0.08714 seconds with return value 0
Press any key to continue . . .
```

Fig. 25.2 Input/ Output screenshot of displaying searching of an element using Binary search.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

25.1 What is the Selection Sort?

25.2 What is the Time and Space Complexity of Selection Sort?

25.3 Can Selection Sort be considered stable or unstable?

25.4 What happens if there are duplicate elements in the array during Selection Sort?

25.5 Can Selection Sort be used for large datasets?

25.6 What is a search algorithm and what types are there?

25.7 What is a binary search?

25.8 What is the difference between binary search and linear search?

25.10 What is the process of binary search?

25.11 What are the drawbacks of binary search?

Experiment No. 26

Objective / Aim: Write a program to add and multiply two matrices(mxn).

Learning Outcomes:

LO1: Apply rules of row/column operations for addition and multiplication.

LO2: Implement matrix addition and multiplication for nxn matrices.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory:

Matrix addition in C is a binary operation that adds two matrices of the same size to produce a new matrix. The new matrix's elements are the sum of the corresponding elements in the original matrices.
Matrix multiplication in C is the process of multiplying two matrices to create a new matrix that represents the combination of the original two. Here are some things to know about matrix multiplication in C:

Conditions:

Matrix multiplication can only be performed if the number of columns in the first matrix is equal to the number of rows in the second matrix.

Result

The resulting matrix will have the same number of rows as the first matrix and the same number of columns as the second matrix.

Order

Matrix multiplication is not commutative, meaning the order of the matrices matters.

Algorithm (Matrix Addition):

```
Step 1: Start
Step 2: Declare matrix mat1[row][col];
        and matrix mat2[row][col];
        and matrix sum[row][col]; row= no. of rows, col= no. of columns
Step 3: Read row, col, mat1[][] and mat2[][]
Step 4: Declare variable i=0, j=0
Step 5: Repeat until i< row
    5.1: Repeat until j < col
            sum[i][j]=mat1[i][j] + mat2[i][j]
            Set j=j+1
    5.2: Set i=i+1
Step 6: sum is the required matrix after addition
Step 7: Stop
```

Flowchart:

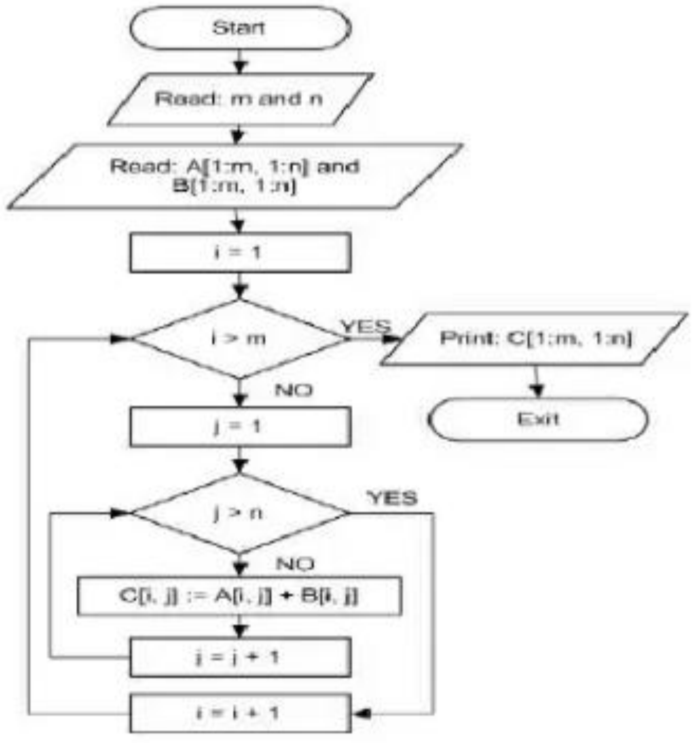
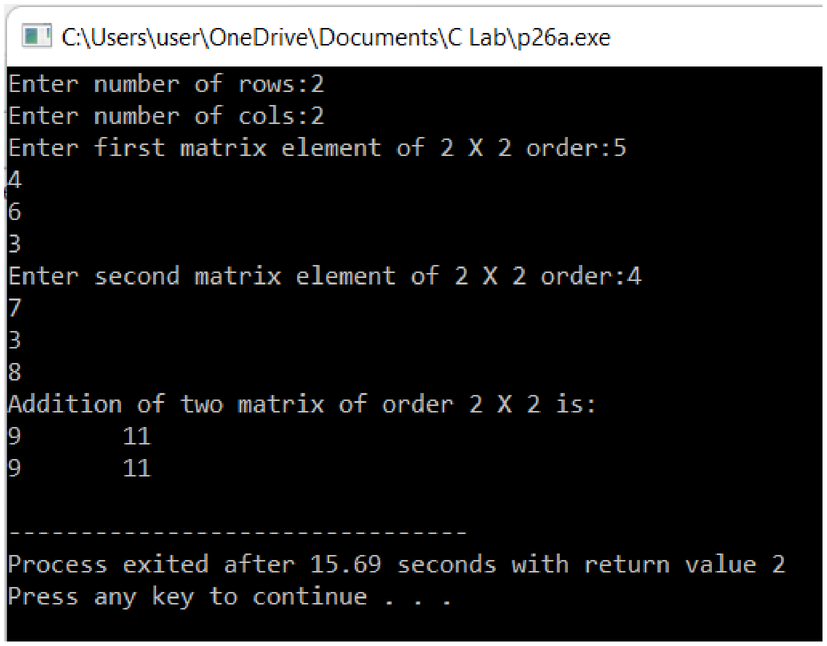


Fig. 26.1 Flowchart of matrix addition.

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p26a.exe
Enter number of rows:2
Enter number of cols:2
Enter first matrix element of 2 X 2 order:5
4
6
3
Enter second matrix element of 2 X 2 order:4
7
3
8
Addition of two matrix of order 2 X 2 is:
9       11
9       11

--------------------------------
Process exited after 15.69 seconds with return value 2
Press any key to continue . . .
```

Fig. 26.2 Input/Output of displaying matrix addition

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Algorithm (Matrix Multiplication):

- **Step 1:** Start
- **Step 2:** Declare matrix mat1[row1][col1]; and matrix mat2[row2][col2]; and matrix mul[row1][col2]; row= no. of rows, col= no. of columns
- **Step 3:** Read mat1[row1][col1] and mat2[row2][col2]
- **Step 4:** Declare variable i=0, j=0
- **Step 5:** Set a loop from i=0 to i=row1.
- **Step 6:** Set an inner loop for the above loop from j=0 to j=col2
    - Initialise the value of the element (i, j) of the new matrix (mul[row1][col2]) to 0.
    - Set an inner loop inside the above loop from k=0 to k=row1.
    - Using the add and assign operator (+=) store the value of mat1[i][k] * mat2[k][j] in the third matrix, mul[i][j].
    - Print the third matrix.
- **Step 7:** Stop

Flowchart:

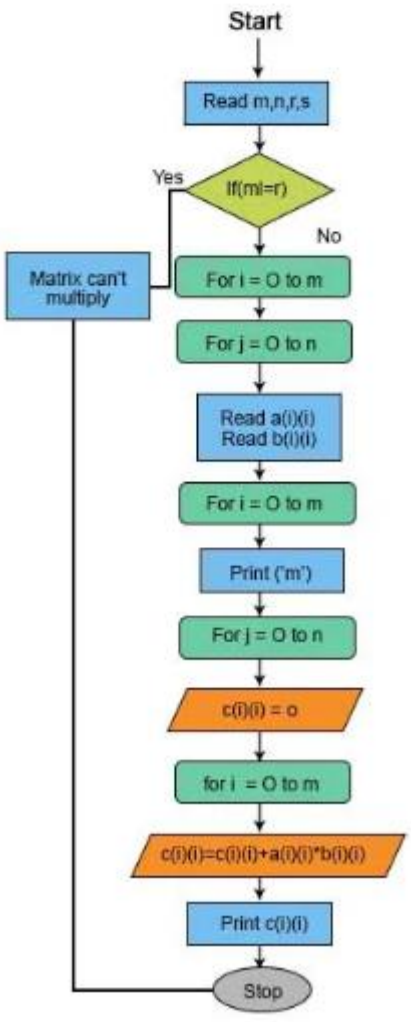
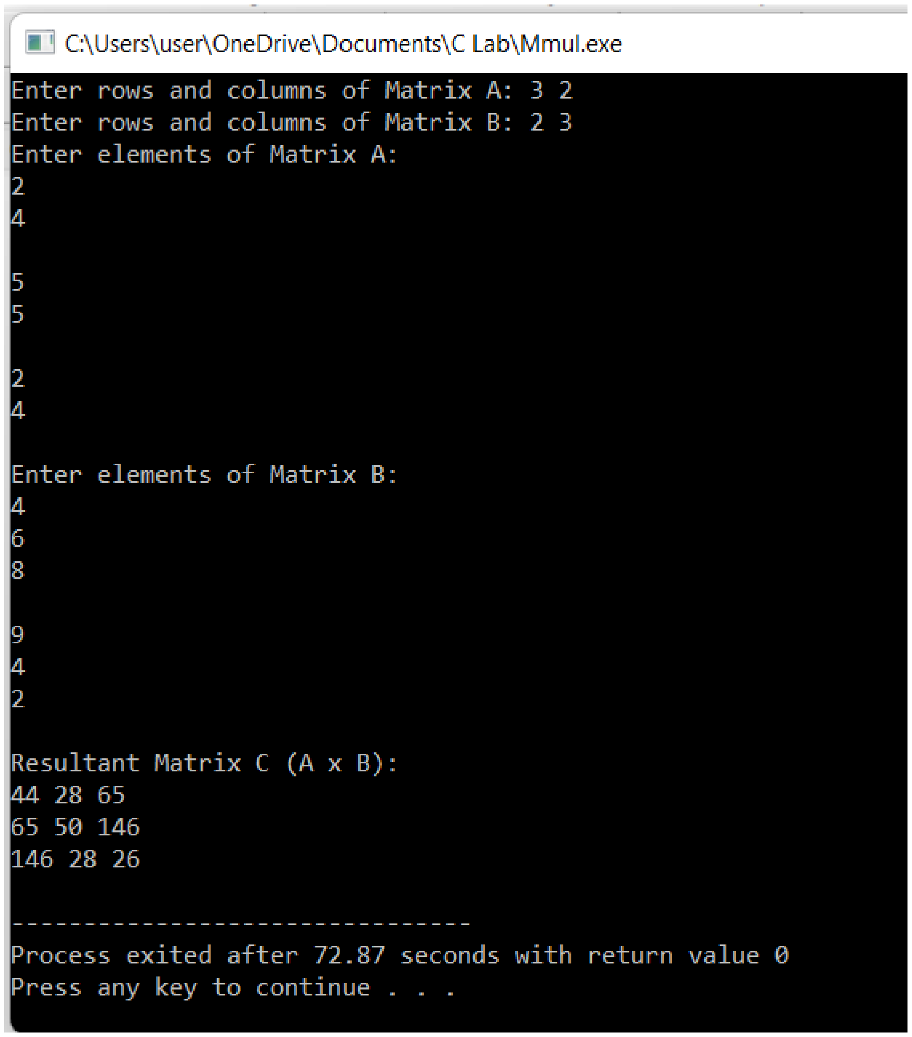


Fig. 26.3 Flowchart of matrix multiplication.

Input/Output(Screenshot):



C:\Users\user\OneDrive\Documents\C Lab\Mmul.exe

```
Enter rows and columns of Matrix A: 3 2
Enter rows and columns of Matrix B: 2 3
Enter elements of Matrix A:
2
4

5
5

2
4

Enter elements of Matrix B:
4
6
8

9
4
2

Resultant Matrix C (A x B):
44 28 65
65 50 146
146 28 26

--------------------------------
Process exited after 72.87 seconds with return value 0
Press any key to continue . . .
```

Fig. 26.4 Input/Output of displaying matrix multiplication.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

26.1 Can matrices of different sizes be added?

26.2 Is matrix addition commutative?

26.3 How does matrix addition work?

26.4 What is matrix multiplication?

26.5 What is the order of the product of two matrices?

26.6 What is the fastest algorithm for matrix multiplication?

Experiment No. 27

Objective / Aim: Write a program that finds the sum of diagonal elements of an m x n matrix.

Learning Outcomes:
LO1: Identify diagonals in a matrix.
LO2: Compute the sum of principal and secondary diagonals.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory: The sum of the diagonal elements of a matrix is called the trace of the matrix. The trace of a matrix (A) is denoted as tr(A). For example, if (A) is an (n x n) matrix, then $tr(A)=a_{11}+a_{22}+\ldots\ldots +a_{nn}$.

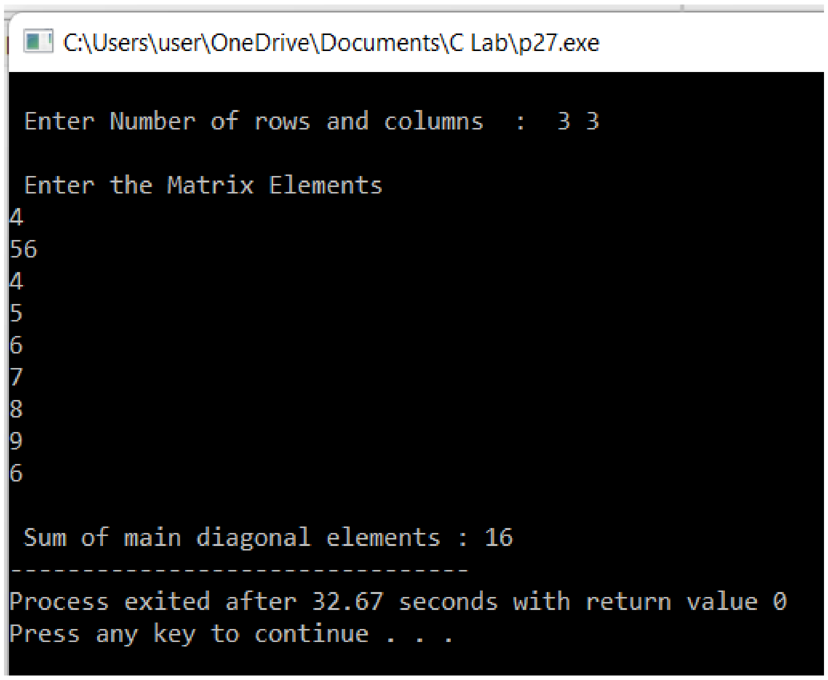
Algorithm (Sum of diagonal elements):

Step1: Start

Step 2: Declare two integer variables, ‘row’ and ‘col’
Step 3: Read the size of the matrix in variables ‘row’ and ‘col’
Step 4: Check if ‘row’ != ‘col’. If this condition is true, display a warning message saying “Please Enter a Square Matrix !!!” and move to step 2.

Step 5: Declare a two-dimensional matrix ‘arr[row][col]’ and a integer variable ‘sum’
Step 6: Run a for loop from ‘i’ = ‘0’ to ‘row’ and ‘j’ = ‘0’ to ‘col’
Step 7: Read the array elements from the user using the ‘scanf’ function.
Step 8: To add the elements of the main diagonal, Run a nested for loop from ‘i’ = ‘0’ to ‘row’
Step 9: Add elements at (‘i’,’i’) using ‘sum’ = ‘sum’ + ‘ arr[i][i]’
Step 10: Return ‘sum’.
Step11: End.

Input/Output(Screenshot):



C:\Users\user\OneDrive\Documents\C Lab\p27.exe

```
 Enter Number of rows and columns  :  3 3

 Enter the Matrix Elements
4
56
4
5
6
7
8
9
6

 Sum of main diagonal elements : 16
--------------------------------
Process exited after 32.67 seconds with return value 0
Press any key to continue . . .
```

Fig. 27.1 Input/Output screenshot of displaying sum of diagonal elements of a matrix.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

27.1 What is trace matrix?

27.2 What is the procedure of sum of diagonal of a matrix?

Experiment No. 28

Objective / Aim: Write a program to print the Fibonacci series and find the factorial of a number using recursion.
Learning Outcomes:
LO1: Understand the concept of recursion and its role in solving problems like factorial and the Fibonacci series.
LO2: Implement recursive functions to generate the Fibonacci series and compute the factorial of a number.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory:

In C programming, recursion is a technique that allows a function to call itself repeatedly to solve a problem. It's similar to breaking down a big problem into smaller, more manageable problems, solving each smaller problem, and then combining the solutions to solve the big problem.

Here are some key concepts of recursion in C:

Recursive calls: When a program allows a user to call a function from within the same function, it's called a recursive call.

Base case: The simplest case that can be solved without calling the function again. The base case is important because it stops the function from calling itself forever.

Recursive step: The step used to break down the problem into sub-problems.

Here is the diagram showing the recursive call done by the function to find the Fibonacci series:

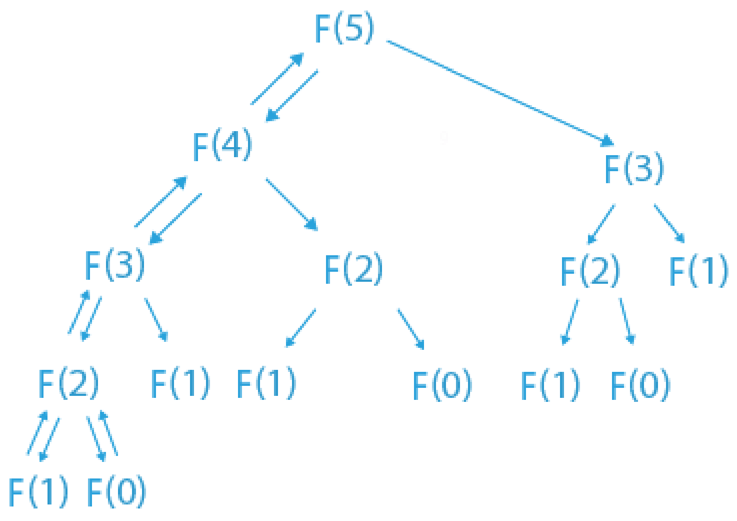


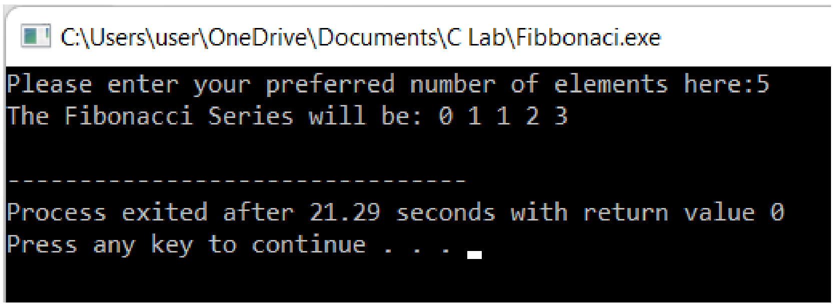
Fig. 28.1 Tree of recursive calling by the function.

Algorithm(Fibonacci Series):

1. Add integer value to num.
2. Then run a for loop from (i = 0; i< num; i++).
3. In each iteration print and call the fibonacciSeries function with i as a parameter.
4. In the Recursive function fibonacciSeries,

- Check if i<= 1, if it is True then return i
- Else return fibonacci(i - 1) + fibonacci(i - 2).
-

Input/Output(Screenshot):


C:\Users\user\OneDrive\Documents\C Lab\Fibbonaci.exe

```
Please enter your preferred number of elements here:5
The Fibonacci Series will be: 0 1 1 2 3

--------------------------------
Process exited after 21.29 seconds with return value 0
Press any key to continue . . .
```

Fig. 28.2 Input/Output screenshot of displaying fibonacci series up to n number.

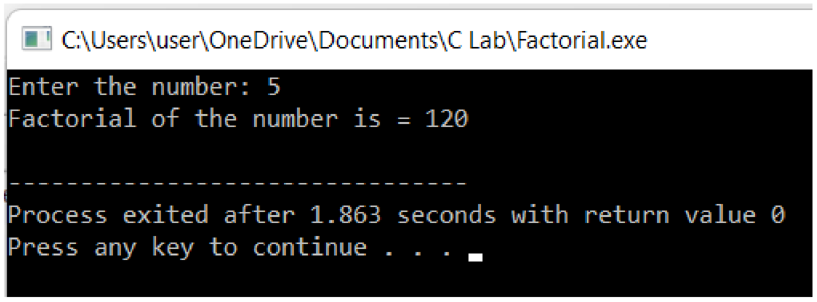
Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Algorithm (Factorial of a number):

```
Step 1: Start
Step 2: Read number n
Step 3: Call factorial(n)
Step 4: Print factorial f
Step 5: Stop
factorial(n)
Step 1: If n==1 then return 1
Step 2: Else
        f=n*factorial(n-1)
Step 3: return f
```

Input/Output(Screenshot):



C:\Users\user\OneDrive\Documents\C Lab\Factorial.exe

```
Enter the number: 5
Factorial of the number is = 120

--------------------------------
Process exited after 1.863 seconds with return value 0
Press any key to continue . . .
```

Fig. 28.3 Input/Output screenshot of displaying factorial of a number.

Precautions and Errors:

- Follow the proper syntax of c programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

28.1 Can you explain the concept of recursion?

28.2 How do you determine the base case in a recursive function?

28.3 How do you avoid infinite recursion in a recursive function?

Experiment No. 29

Objective : Write a program to implement the string library functions trlen(), strcpy(), strcat(), strcmp()**.**

Learning Outcomes:
LO1: Understand the logic of string length, concatenation, and copy.

LO2: Implement user-defined string functions using loops and functions

Apparatus Used: Turbo c compiler, gcc compiler (vs code, dev c)

Theory:

Library functions are built-in functions that are grouped together and placed in a common location called library. Each function here performs a specific operation. We can use this library functions to get the pre-defined output.

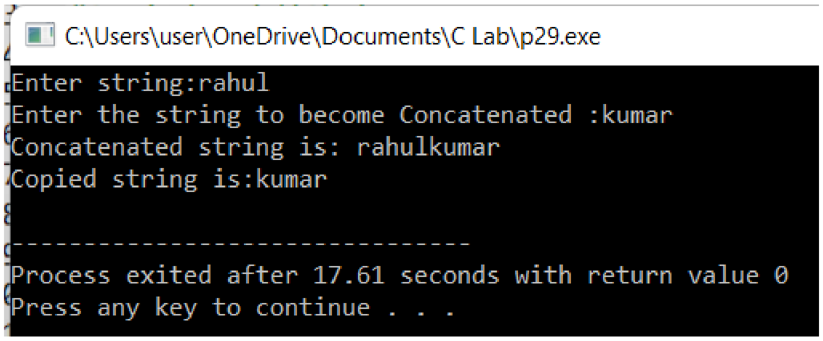
All C standard library functions are declared by using many header files.These library functions are created atthetime of design in the compilers.

We include the header files in our C program byusing #include<filename.h> .Whenever the program is run and executed, the related files are included in the C program.

Algorithm:

```
Step1: Start
Step2: Read string [50], dest [50],c1[50]
Step3:Print the length of string is strlen(string)
Step4: strcpy (dest,string)
Step 5: print the value at source is string
Step6:print th evalueatd estination is dest
Step7:strcat(string,c1)
Step8:print the concatenated string isstring
Step9:stop
```

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p29.exe
Enter string:rahul
Enter the string to become Concatenated :kumar
Concatenated string is: rahulkumar
Copied string is:kumar

--------------------------------
Process exited after 17.61 seconds with return value 0
Press any key to continue . . .
```

Fig. 29.1 Input/Output screenshot of displaying concatenation of two strings.

Precautions and Errors:

- Follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

29.1 What does the function strlen() do in C?

29.2 What is the difference between strcpy() and strncpy()?

29.3 What is the difference between printf() and puts() in C?

Experiment No. 30

Objective : WAP to find the minimum and maximum element of the array.

Learning Outcomes (LOs):

LO1: Understand the concept of array traversal to access elements sequentially.

LO2: Implement a program to identify and display the minimum and maximum elements of an array.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature/Theory/Formula:

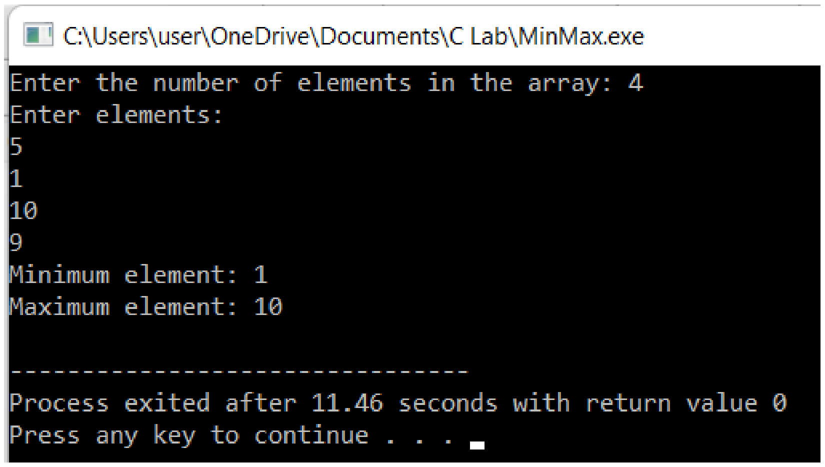
An array in C is a fixed-size collection of similar data it stored in contiguous memory locations. It is used to store multiple values under asingle name ,making it easier to manage and manipulate large sets of data. Arrays can store primitive data type such as int, char, float, and also derived and user-defined data types like pointers and structures.

Algorithm:

```
1. Start
2. Input Size: Read the size of the array n.
3. CheckValidity: Ifn<=0,print an error message and exit.
4. Input Elements:
    a. Declare an array arr of size n.
    b. For i= 0ton-1,do:
        i. Read arr [i].
5. Initialize Min and Max:
    a. Set min=arr[0](the first element).
    b. Set max=arr[0](the first element).
6. Iterate Through the Array:
    a. For i= 1ton-1,do:
        i. I farr[i]>max, then:
            1. Set max= arr[i](update maximum).
        ii. Ifarr[i]<min,then:
            1. Se tmin=arr[i](update minimum).
7. Output Results:
    a. Print min(the minimum value).
```

    b. Print max(the maximum value).
8. End

Input/Output(Screenshot):


C:\Users\user\OneDrive\Documents\C Lab\MinMax.exe

```
Enter the number of elements in the array: 4
Enter elements:
5
1
10
9
Minimum element: 1
Maximum element: 10

--------------------------------
Process exited after 11.46 seconds with return value 0
Press any key to continue . . .
```

Fig. 30.1 Input/Output screenshot of displaying Min and Max elements of an array.

Precautions and Errors:

- Follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

30.1 What is an array in C?

30.2 How do you declare an array in C?

30.3 How can you initialize an array in C?

30.4 How do you access an element of an array?

Experiment No. 31

Objective: Define a structure data type name STUDENT. The type contains student id: string type, student name: type string, student age: type integer, total marks: float type. Display all the record of the student.

Learning Outcomes:

LO1: Store train details using a structure.

LO2: Retrieve and filter train records by station and time.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory:

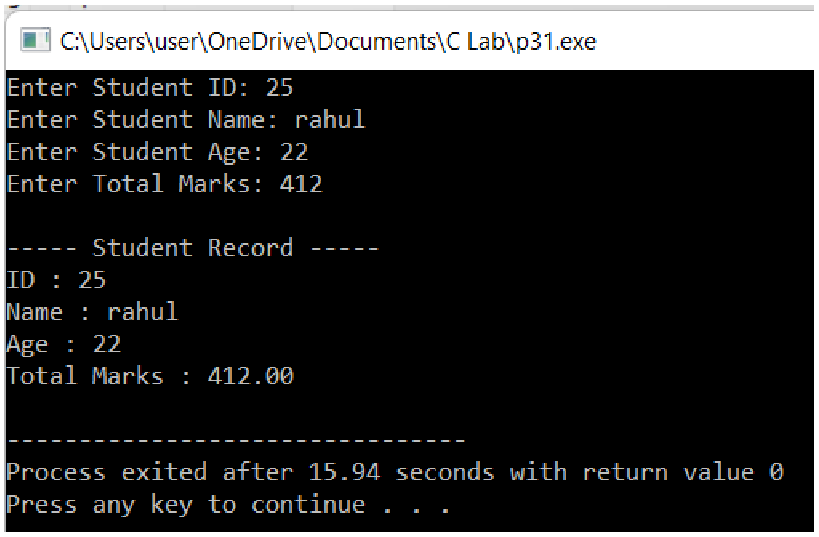
Structure in C is a user-defined data type that enables us to store the collection of different data types. Each element of a structure is called a member. Structures in C; simulate the use of classes and templates as it can store various in formation
The **struct** key word is used to define the structure. Let's see the syntax to define the structure in C.

Algorithm:

1. Define the structure STUDENT with thefields:student_id,student_name,student_age,andtotal_marks.
2. Declare an array of STUDENT to hold multiple records.
3. Prompt the user to enter the number of students (upto a maximum).
4. If the number of students is invalid (≤0orexceedsmaximum), display an error message and exit.
5. For each student:

   Call input Student Details() to get the details from the user.
6. After entering all student records, display each student's details using display Student Details().
7. End the program.

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p31.exe
Enter Student ID: 25
Enter Student Name: rahul
Enter Student Age: 22
Enter Total Marks: 412

----- Student Record -----
ID : 25
Name : rahul
Age : 22
Total Marks : 412.00

--------------------------------
Process exited after 15.94 seconds with return value 0
Press any key to continue . . .
```

Fig. 31.1 Input/Output screenshot of displaying record of a student using structure data type.

Precautions and Errors:

- Follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

31.1 How do you initialize a structure in C?

31.2 Can you initialize only some members of a structure?

31.3 How do you access members of a structure in C?

Experiment No. 32

Objective: Define a structure data type name STUDENTS.The type contains student id:string type, student name: type string, student age: type integer, total marks: float type. Display all the record of the n students using array of structure.

Learning Outcomes:

LO1: Define and store student details using structure.

LO2: Display all student records.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory:

Structure in C is a user-defined data type that enables us to store the collection of different data types. Each element of a structure is called a member. Structure in C a; simulate the use of classes and templates as it can store various information The **struct** keyword is used to define the structure. Let's see the syntax to define the structure in C.

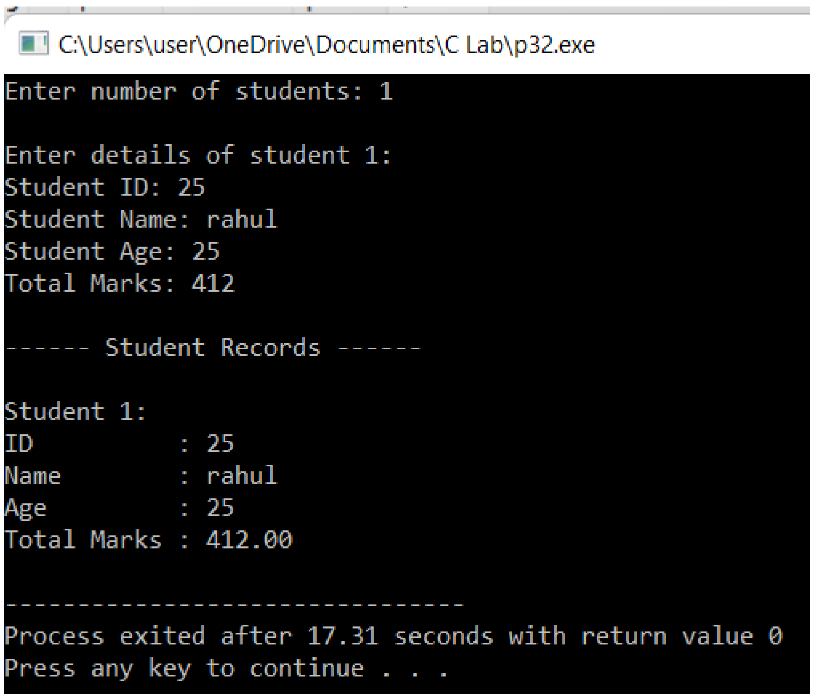
Algorithm:

1. Define the Structure:
   - Define the STUDENTS structure with the following fields:
     - student_id:string type
     - student_name:string type
     - student_age:integer type
     - total_marks:float type
2. Declare an Array:
   - Declare anarray of STUDENTS to hold multiple records.
3. Input Number of Students:
   - Prompt the user to enter the number of students (up to a maximum defined).
4. Check Validity:
   - If the number of students is invalid (≤0orexceedsmaximum), display an error message and exit.
5. Input Student Details:
   - For each student:
     - Call input Student Details()to get the details from the user.
6. Display Student Records:
   - After entering all student records, display each student's details using

display Student Details ().

7. End the Program:

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p32.exe
Enter number of students: 1

Enter details of student 1:
Student ID: 25
Student Name: rahul
Student Age: 25
Total Marks: 412

------ Student Records ------

Student 1:
ID          : 25
Name        : rahul
Age         : 25
Total Marks : 412.00

--------------------------------
Process exited after 17.31 seconds with return value 0
Press any key to continue . . .
```

Fig. 32.1 Input/Output screenshot of displaying record of a student using structure data type.

Precautions and Errors:

- Follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

32.1 What is an array of structure ?

Experiment No. 33.

Objective/Aim: Create structure data type name ADDRESS;-The type contains city: string type, pin-code: type integer. Create structure data type name EMPLOYEE;- The type contains name: string type. Display all the information of the Employee using concept of nested structure.

Learning Outcomes:
LO1: Define and use nested structures.
LO2: Store and display employee details including address.

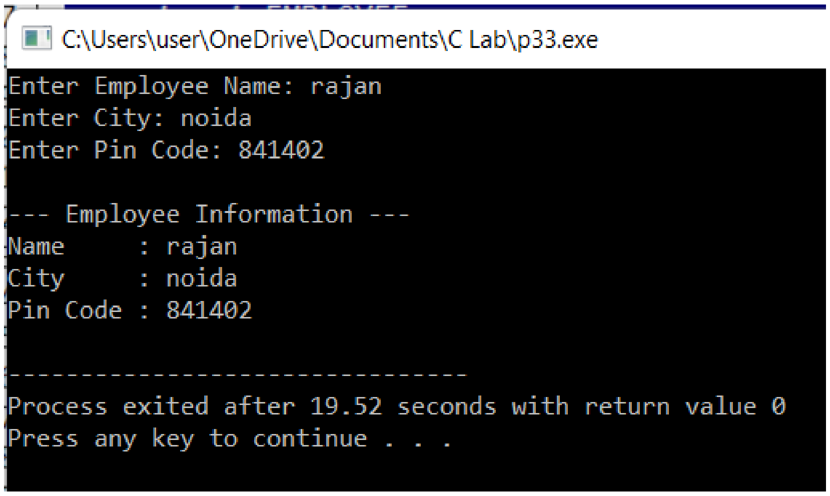
Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature /Theory/Formula:
Structure in C is a user-defined data type that enables us to store the collection of different data types.
Each element of a structure is called a member. Structures in C; simulate the use of classes and templates as it can store various information
The **struct** key word is used to define the structure. Let's see the syntax to define the structure in C

Algorithm:
Define the Structures:
Define the ADDRESS structure with the following fields: city: string type, pin code: Integer type
Define the EMPLOYEE structure with the following fields:
name: string type
address: nested structure of type ADDRESS
Declare an Employee Variable:
Declare a variable of type EMPLOYEE to hold the employee's details.
Input Employee Details:
Call inputEmployeeDetails() to get the employee's name ,city, and pin-code.
Display Employee Information:
Call display EmployeeDetails() to print the employee's name and address.
End the Program:

Exit the program gracefully.

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p33.exe
Enter Employee Name: rajan
Enter City: noida
Enter Pin Code: 841402

--- Employee Information ---
Name      : rajan
City      : noida
Pin Code : 841402

--------------------------------
Process exited after 19.52 seconds with return value 0
Press any key to continue . . .
```

Fig. 33.1 Input/Output screenshot of displaying information of an employee using structure data type.

Precautions and Errors:

1. Follow the proper syntax of C programming.

2. Declare the data type as per the problem statement

Questions for Viva-voice:

33.1 What happens when a structure is passed to a function by value?

33.2 Can a structure contain another structure?

33.3 How do you access members of a nested structure?

Experiment No. 34

Objective/Aim: Write a program to Swap Two Elements Using Pointers.

Learning Outcomes:

LO1: Understand the use of pointers in C.

LO2: Implement swapping using pointer referencing.

LO3: Explain the benefits of pointers in function parameter passing.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature/Theory/Formula:

Pointers are one of the core components of the C programming language. A pointer can be used to store the **memory address** of other variables, functions, or even other pointers. The use of pointers allow slow-level memory access, dynamic memory allocation in the functionality in C.

Algorithm:

Define the Function:

- Create a function named swap that accepts two integer pointers as parameters.
- This allows the function to access and modify the values at the memory locations the pointers refer to.

Create a Temporary Variable:

- Inside the swap function, declare at temporary integer variable named temp.
- This variable will temporarily hold the value of one of the integers during the swap process.

Storethe Value of the First Integer:

- Assign the value pointed to by a to temp. This preserves the value of the first integer so that it can be assigned to the second integer later.

Assign the Value of the Second Integer:

- Update the value at the address pointed to by a with the value at the address pointed.
- Thiseffectivelychangesthefirstinteger'svaluetothesecondinteger'svalue.

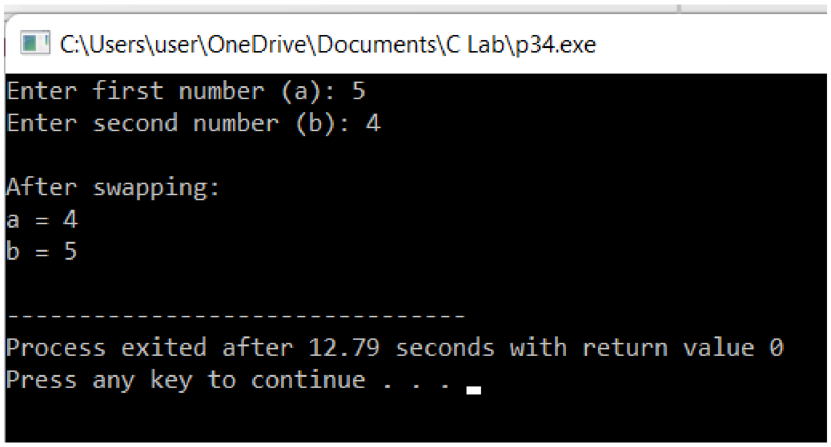
Assign the Temporary Value to the Second Integer:

- Finally, assign the value stored in temp (the original value of the first integer) to the location pointed.
- This completes swap.

Return to the Calling Function:

- Once the swapping is done, return control back to the calling function (main progra

Input/Output(Screenshot):


C:\Users\user\OneDrive\Documents\C Lab\p34.exe

```
Enter first number (a): 5
Enter second number (b): 4

After swapping:
a = 4
b = 5

--------------------------------
Process exited after 12.79 seconds with return value 0
Press any key to continue . . .
```

Fig. 34.1 Input/Output screenshot of displaying swapping of two numbers using a pointer.

Precautions and Errors:

- follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

34.1 What happens when a structure is passed to a function by value?

34.2 Can a structure contain another structure?

34.3 How do you access members of a nested structure?

Experiment No. 35

Objective: WAP to check whether a given word exists in a file or not. If yes, then find the number of times it occurs.

Learning Outcomes:

LO1: Read text data from a file.

LO2: Search for a given word and count occurrences.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory:

This program reads the contents of a file and searches for the presence of a specific word provided by the user. If the word is found, the program counts how many times it appears in the file.

Algorithm:

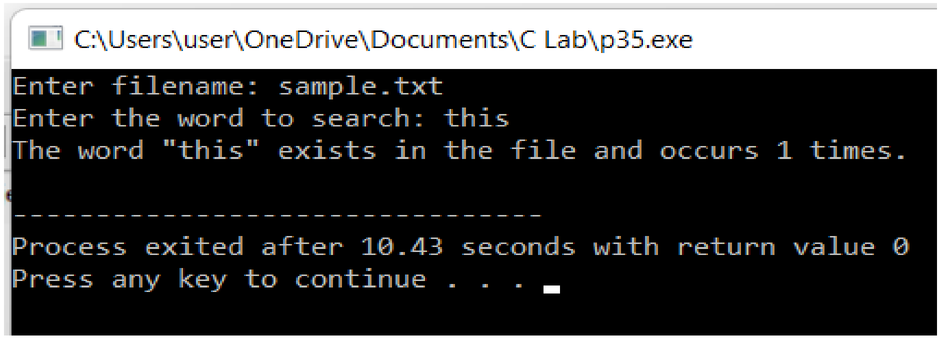
1. Start the program.
2. Open the file in read mode.
3. If the file cannot be opened, print an error and exit.
4. Ask the user to input the word to be searched.
5. Read the file word by word using `fscanf`.
6. For each word:
    - Compare it with the input word using `strcmp`.
    - If they match, increment a counter.
7. After reading the file, check the counter:
    - If > 0, print the number of times the word was found.
    - Else, print that the word was not found.
8. Close the file.
9. End the program.

User Input
Contents of sample.txt:
This is a sample file. It contains the word file several times.

Input:

Enter the word to search: This

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p35.exe
Enter filename: sample.txt
Enter the word to search: this
The word "this" exists in the file and occurs 1 times.

--------------------------------
Process exited after 10.43 seconds with return value 0
Press any key to continue . . .
```

Fig. 35.1 Input/Output screenshot of displaying total count of a token in a file.

Precautions and Errors:

- follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

35.1 What is fscanf() used for?

35.2 What does strcmp() do?

35.3 How do you define a word in this program?

Experiment No. 36

Objective/Aim: WAP to compare the contents of two files and determine whether they are same or not.

Learning Outcomes:

LO1: Read the contents of two files.

LO2: Implement comparison logic to check if files are identical.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Literature/Theory/Formula:

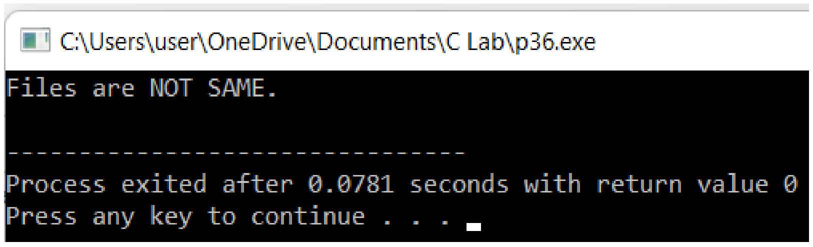
File comparison is the process of checking whether two files have identical contents. This is useful in many applications like version control, data backup verification, or detecting tampered files.

Algorithm:

1. Start the program.
2. Open both files in read mode.
3. If any of the files cannot be opened, display an error and exit.
4. Read characters from both files using a loop.
5. Compare characters one by one.
6. If a mismatch is found, display that files are **not the same** and exit the loop.
7. If end of both files is reached together without mismatches, display that files are **same**.
8. Close both files.
9. End the program.

User Input
Contents of file1.txt:
Hello World!
Contents of file2.txt:
World!

Output:



```
Files are NOT SAME.

--------------------------------
Process exited after 0.0781 seconds with return value 0
Press any key to continue . . .
```

Fig. 36.1 Output screenshot of displaying two files are the same or not.

Precautions and Errors:

- Follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

36.1 What is the purpose of fgetc() in C?

36.2 What does EOF stand for?

.36.3 Why do we need to compare files?

Experiment No. 37

Objective: WAP to check whether the given no is a Palindrom or not.

Learning Outcomes:

LO1: Reverse the digits of a number.

LO2: Check if the number equals its reverse.

LO3: Extend to string palindromes and real-life scenarios.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

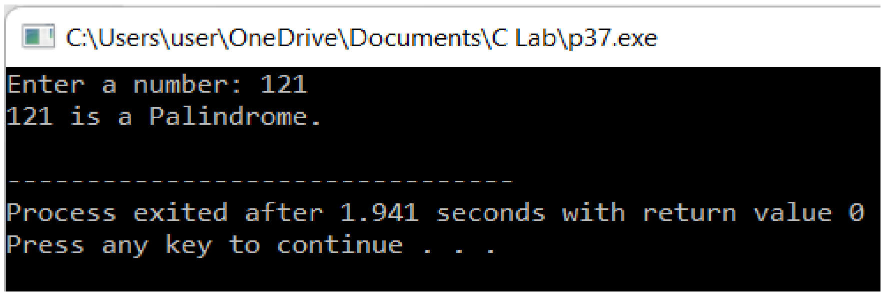
Theory:

A number is called a palindrome if it remains the same when its digits are reversed. For example, 121 and 1331 are palindromes.

Algorithm:

1. Start
2. Input a number
3. Initialize a variable to store the reversed number
4. Extract each digit and build the reversed number
5. Compare original and reversed numbers
6. Print result
7. Stop

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p37.exe
Enter a number: 121
121 is a Palindrome.

--------------------------------
Process exited after 1.941 seconds with return value 0
Press any key to continue . . .
```

Fig. 37.1 Input/Output screenshot of displaying a string is palindrome or not.

Precautions and Errors:

- follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

37.1 What is a palindrome number?

37.2 Which logic is used to reverse a number in this program?

Experiment No. 38

Objective: WAP to check whether the no is Perfect or not.

Learning Outcomes:

LO1: Identify factors of a number.

LO2: Sum the factors and check if it equals the number.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

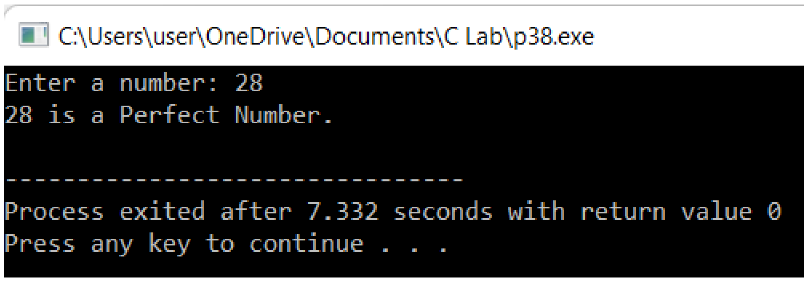
Theory:

A perfect number is a positive integer that is equal to the sum of its proper divisors (excluding itself). Example: 6 = 1 + 2 + 3

Algorithm:

1. Start
2. Input a number
3. Initialize sum = 0
4. Loop from 1 to n-1
5. If n % i == 0, add i to sum
6. Compare sum and original number
7. Display result
8. Stop

Input/Output(Screenshot):



```
Enter a number: 28
28 is a Perfect Number.

--------------------------------
Process exited after 7.332 seconds with return value 0
Press any key to continue . . .
```

Fig. 38.1 Input/Output Screenshot of displaying a Perfect number.

Precautions and Errors:

- Follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

38.1 What is a perfect number?

38.2 Which loop is used to find the divisors of the number?

Experiment No. 39

Objective: WAP to make Hotel Menu using Switch Case.

Learning Outcomes:

LO1: Understand switch-case statement.

LO2: Implement a menu-driven program for hotel system.

LO3: Relate switch-case menus to user-interface design.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

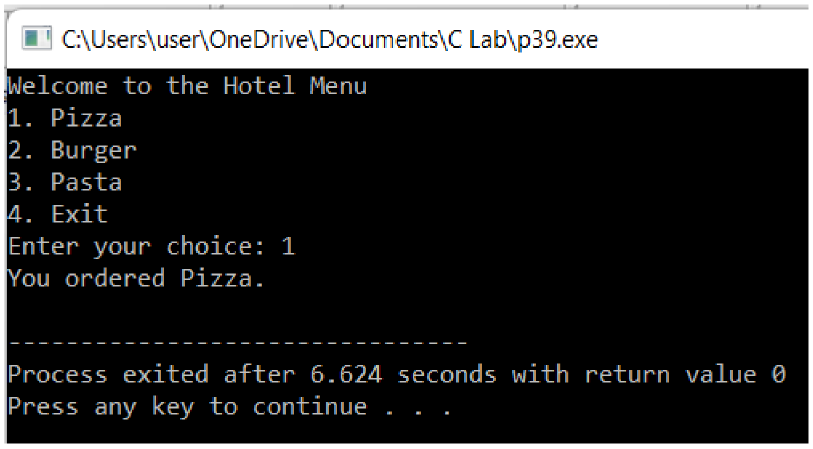
Theory:

Switch-case allows branching based on the value of a variable. It is used for menu-driven programs.

Algorithm:

1. Start
2. Display menu
3. Read user choice
4. Use switch-case to print selected item
5. Stop

Input/Output(Screenshot):



C:\Users\user\OneDrive\Documents\C Lab\p39.exe

```
Welcome to the Hotel Menu
1. Pizza
2. Burger
3. Pasta
4. Exit
Enter your choice: 1
You ordered Pizza.

--------------------------------
Process exited after 6.624 seconds with return value 0
Press any key to continue . . .
```

Fig. 39.1 Input/Output Screenshot of displaying a hotel menu using Switch Case.

Precautions and Errors:

- Follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

39.1 Why do we use a switch-case in this program?

39.2 What is the purpose of the default case in a switch?

Experiment No. 40

Objective: WAP to Sum Two Numbers Without Using '+' Operator.

Learning Outcomes:

LO1: Understand addition using bitwise operations.

LO2: Implement bitwise-based addition logic.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

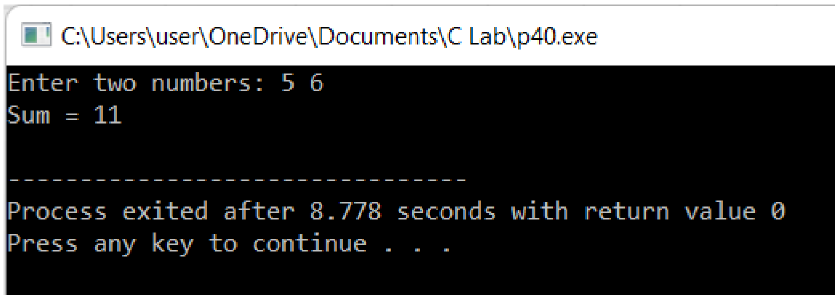
Theory:

Bitwise operators can be used to add numbers without the '+' operator. XOR gives sum without carry, AND followed by left shift gives carry.

Algorithm:

1. Start
2. Input two integers
3. Repeat while carry is non-zero
4. Use bitwise operations to compute sum
5. Print result
6. Stop

Input/Output(Screenshot):



```
C:\Users\user\OneDrive\Documents\C Lab\p40.exe
Enter two numbers: 5 6
Sum = 11

--------------------------------
Process exited after 8.778 seconds with return value 0
Press any key to continue . . .
```

Fig. 40.1 Input/Output Screenshot of displaying sum of two numbers without using "+".

Precautions and Errors:

- Follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

40.1 How do we add two numbers without using +?

40.2 Why is the loop used in bitwise addition?

Experiment No. 41

Objective: WAP to Print Right and Left Pyramid Pattern.

Learning Outcomes:

LO1: Use loops to generate shapes.

LO2: Implement logic to print right and left pyramid.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

Theory:

Patterns are printed using nested loops where rows and columns control the shape and spacing of characters.

Algorithm:

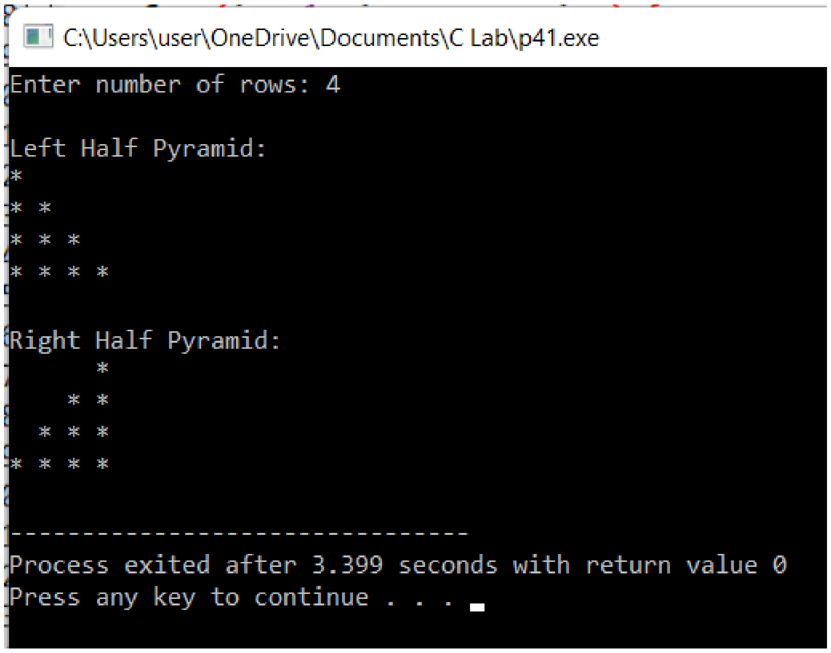
Right Half Pyramid Algorithm:

1. Start
2. Input rows
3. Outer loop for rows
4. Inner loop for columns
5. Print '*'
6. Stop

Left Half Pyramid Algorithm:

1. Start
2. Input rows
3. Outer loop for rows
4. Print spaces
5. Print '*'
6. Stop

Input/Output(Screenshot):



C:\Users\user\OneDrive\Documents\C Lab\p41.exe

```
Enter number of rows: 4

Left Half Pyramid:
*
* *
* * *
* * * *

Right Half Pyramid:
      *
    * *
  * * *
* * * *

--------------------------------
Process exited after 3.399 seconds with return value 0
Press any key to continue . . .
```

Fig. 41.1 Input/Output Screenshot of displaying left and right pyramid.

Precautions and Errors:

- Follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

41.1 Which loop structure is used for printing pyramid patterns?

41.2 How do we align the left pyramid pattern?

Experiment No. 42

Objective: WAP to Count Alphabets, Digits, and Special Characters.

Learning Outcomes:

LO1: Identify alphabets, digits, and symbols using ASCII values.

LO2: Implement a program to count each type in a string.

Apparatus Used: Turbo C compiler, GCC compiler (VS Code, Dev C).

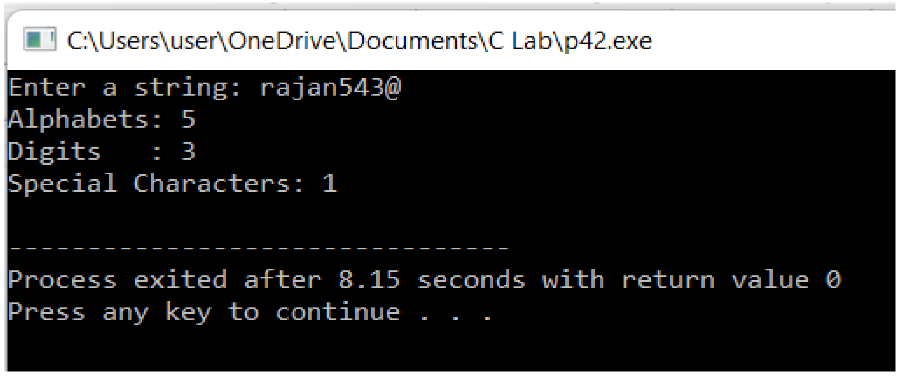
Theory:

Characters in a string can be classified using their ASCII values. Alphabets (A-Z, a-z), Digits (0-9), and others are special characters.

Algorithm:

1. Start
2. Input a string
3. Loop through each character
4. Check if character is alphabet, digit, or special
5. Increment corresponding counter
6. Display counts
7. Stop

Input/Output(Screenshot):



C:\Users\user\OneDrive\Documents\C Lab\p42.exe

```
Enter a string: rajan543@
Alphabets: 5
Digits   : 3
Special Characters: 1

--------------------------------
Process exited after 8.15 seconds with return value 0
Press any key to continue . . .
```

Fig. 42.1 Input/Output Screenshot of displaying counts of numbers, characters in a string.

Precautions and Errors:

- Follow the proper syntax of C programming
- Declare the data type as per the problem statement.

Questions for Viva-voice:

42.1 How do we check if a character is an alphabet, digit, or special character?

42.2 Which function can be used instead of ASCII comparisons?