

Programming Fundamentals

(BCL 103)

Lab Practical Report



Faculty name - Deepika Garg

Student name - Aryan Yadav

Roll No.: 24BCA004

Semester: First

Group: A (First)

Department of Computer Science and Engineering

NorthCap University, Gurugram- 122001, India

Session 2024-25

Experiment- 1

Student Name and Roll Number: Aryan Yadav (24bca004)
Semester /Section: First Sem.
Link to Code:
Date: 24/10/2024
Faculty Signature:

Objective

To familiarize the students about Number Systems

Program Outcome

- The students will be able to represent numbers in various numeration systems , convert a numeral from one base to another base and perform arithmetic operations in bases other than ten

Background Study:

A number system is defined as a system of writing for expressing numbers. It is the mathematical notation for representing numbers of a given set by using digits in a consistent manner. It provides a unique representation of every number and represents the arithmetic and algebraic structure of the figures. It also allows us to operate arithmetic operations like addition, subtraction, and division.

There are various types of the number system in mathematics. The four most common number system types are:

System	Base	Symbols
--------	------	---------

Decimal	10	0, 1, ... 9
Binary	2	0, 1
Octal	8	0, 1, ... 7
Hexa-decimal	16	0, 1, ... 9, A, B, ... F

Perform below Number system conversions, write all steps with the outcome:

1. Convert $(1056)_{16}$ to $(?)_8$
2. Convert $(11672)_8$ to $(?)_{16}$
3. Convert $(2724)_8$ to $(?)_{10}$
4. Convert $(3211)_4$ to $(?)_5$
5. Convert $(1001001100)_2$ to $(?)_6$

Experiment-2

Student Name and Roll Number: Aryan Yadav (24bca004)
Semester /Section: First Sem.
Link to Code:
Date: 24/10/2024
Faculty Signature:

Objective

To familiarize the students about Number Systems

Program Outcome

- The students will be able to represent numbers in various numeration systems , convert a numeral from one base to another base and perform arithmetic operations in bases other than ten

Background Study:

A number system is defined as a system of writing for expressing numbers. It is the mathematical notation for representing numbers of a given set by using digits in a consistent manner. It provides a unique representation of every number and represents the arithmetic and algebraic structure of the figures. It also allows us to operate arithmetic operations like addition, subtraction, and division.

There are various types of the number system in mathematics. The four most common number system types are:

System	Base	Symbols
Decimal	10	0, 1, ... 9

Binary	2	0, 1
Octal	8	0, 1, ... 7
Hexa-decimal	16	0, 1, ... 9, A, B, ... F

- Convert the following 4-bit numbers from binary to decimal.
 - 0101_2
 - 0111_2
 - 0011_2
 - 1001_2
 - 1011_2
 - 1111_2
 - 0000_2
 - 1101_2
- Convert the following 8-bit numbers from binary to decimal.
 - 00010101_2
 - 10110101_2
 - 11010011_2
 - 01101000_2
- Convert the following 16-bit numbers from binary to decimal.
 - $10110101\ 00010101_2$
 - $01101000\ 11010011_2$
- Determine whether the following statements are true or false.
 - $1001_2 < 5_{10}$
 - $0111_2 = 111_{10}$
 - $0011_2 > 2_{10}$
 - $1001_2 > 1101_2$
 - $1011_2 = 11_{10}$
 - $1111_2 = 15_{10}$
 - $0000_2 < 0_{10}$

h. $1101_2 > 1010_2$

Experiment-3

Student Name and Roll Number: Aryan Yadav (24bca004)
Semester /Section: First Sem.
Link to Code:
Date: 24/10/2024
Faculty Signature:

Objective

To familiarize the students about Number Systems

Program Outcome

- The students will be able to represent numbers in various numeration systems , convert a numeral from one base to another base and perform arithmetic operations in bases other than ten

Background Study:

A number system is defined as a system of writing for expressing numbers. It is the mathematical notation for representing numbers of a given set by using digits in a consistent manner. It provides a unique representation of every number and represents the arithmetic and algebraic structure of the figures. It also allows us to operate arithmetic operations like addition, subtraction, and division.

There are various types of the number system in mathematics. The four most common number system types are:

System	Base	Symbols
Decimal	10	0, 1, ... 9
Binary	2	0, 1

Octal	8	0, 1, ... 7
Hexa-decimal	16	0, 1, ... 9, A, B, ... F

1. What is the decimal equivalent of the number $3A_{16}$?
2. What is the 8 bit unsigned binary result of $56_{10} - 31_{10} = 25$
3. What is the result of adding 7_{10} and -4_{10} using 8 bit signed binary notation?
4. Which of the following 4 bit Excess 3 numbers is equivalent to 5_{10} ?
5. Consider the equation $(123)_5 = (x8)_y$ with x and y as unknown. The number of possible solutions is _____ .
6. Convert binary 11111110010 to hexadecimal.
7. The representation of octal number $(532.2)_8$ in decimal is _____
8. The decimal equivalent of the octal number $(645)_8$ is _____
9. The quantity of double word is _____
10. Octal to binary conversion: $(24)_8 = ?$
11. Convert binary to octal: $(110110001010)_2 = ?$
12. The octal number $(651.124)_8$ is equivalent to (_____) $_{10}$.
13. Convert the hexadecimal number $(1E2)_{16}$ to decimal:
14. Let r denote number system radix. The only value(s) of r that satisfy the equation $\sqrt{121_r} = 11_r$ is/are

Experiment- 4

Student Name and Roll Number: Aryan Yadav (24bca004)
Semester /Section: First Sem.
Link to Code:
Date: 24/10/2024
Faculty Signature:

Objective: Basic C Programming Questions

Program Outcome: Basic c programming understanding and able to perform simple programming questions

Background Study: Input/Output in C can be achieved using scanf() and printf() functions. The printf and scanf are two of the many functions found in the C standard library. These functions are declared and related macros are defined in stdio.h header file. The printf function is used to write information from a program to the standard output device whereas scanf function is used to read information into a program from the standard input device.

Function Prototype of printf and scanf in C

Function name	Function prototype
Printf	int printf(const char* format, ...);
Scanf	int scanf(const char* format, ...);

Format Specifier of printf and scanf Functions

Format specifier	Description
%d	Signed decimal integer
%u	Unsigned decimal integer
%f	Floating point numbers
%c	Character
%s	Character String terminated by '\0'
%p	Pointer address

Q1. Write a C program to add two integer numbers.

Code: #include <stdio.h>

```
int main() {
    int num1, num2, sum;

    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    sum = num1 + num2;

    printf("Sum: %d\n", sum);

    return 0;
}
```

Output and screenshot:

```
Output Clear  
/tmp/EjwNjWME1F.o  
Enter two integers: 10 30  
Sum: 40  
  
=== Code Execution Successful ===
```

Q2. Write a C Program to add two floating numbers.

Code:

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

```
int main() {
```

```
    float num1, num2, sum;
```

```
printf("Enter two floating-point numbers: ");  
  
scanf("%f %f", &num1, &num2);  
  
sum = num1 + num2;  
  
printf("Sum: %.2f\n", sum);  
  
return 0;  
}
```

Output and screenshot:

Output Clear

```
/tmp/TyoTM0HSX3.o  
Enter two floating-point numbers: 10.30 31.2  
Sum: 41.50  
  
=== Code Execution Successful ===
```

Q3. Write a c program to display Hello World!

Code:

```
#include <stdio.h>

int main() {
    printf("Hello World!\n");
    return 0;
}
```

Output and screenshot:

Output Clear

```
/tmp/yfVFRcv8sj.o
Hello World!

=== Code Execution Successful ===
```

Q4. Write a C Program to check if the number entered by the user is even or odd.

Code:

```
#include <stdio.h>

int main() {
    int num;

    printf("Enter an integer: ");
    scanf("%d", &num);

    if (num % 2 == 0)
        printf("%d is even.\n", num);
    else
        printf("%d is odd.\n", num);

    return 0;
}
```

Output and screenshot:

```
Output Clear  
/tmp/och6CLjajP.o  
Enter an integer: 25  
25 is odd.  
  
=== Code Execution Successful ===
```

Q5. Write a program to check whether the character value entered by the user is Vowel or Consonant.

Code:

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

```
int main() {
```

```
    char ch;
```

```
    printf("Enter a character: ");
```

```
    scanf("%c", &ch);
```

```
if (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' ||  
    ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U') {  
    printf("%c is a vowel.\n", ch);  
} else {  
    printf("%c is a consonant.\n", ch);  
}
```

```
return 0;
```

Output and screenshot:

Output Clear

```
/tmp/cTspJN7TwB.o  
Enter a character: A  
A is a vowel.  
  
=== Code Execution Successful ===
```


Experiment - 5

Student Name and Roll Number: Aryan Yadav (24bca004)
Semester /Section: First Sem.
Link to Code:
Date: 24/10/2024
Faculty Signature:

Objective: To study pseudocode and flowcharts basics, symbols, practical usage.

Program Outcome: The various examples of the flowchart

Background Study: A flowchart is a **picture of the separate steps of a process in sequential order**. It is a generic tool that can be adapted for a wide variety of purposes, and can be used to describe various processes, such as a manufacturing process, an administrative or service process, or a project plan. A flowchart is a type of diagram that represents a workflow or process. A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solving a task. The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows

Q1. Design a flowchart for adding two numbers entered by the user.

Q2: Design a flowchart for finding the largest among three numbers entered by the user.

Q3: Design a flowchart for calculating the profit and loss according to the cost price and income value entered by the user.

Q4: Draw a flowchart to calculate the average of two numbers.

Q5: Design a flowchart for the multiplication of three numbers entered by the user.

Q6: Design a flowchart for calculating the area of a rectangle.

Q7: Design a flowchart for calculating Simple Interest according to the value of principal amount, ROI, and time the user enters.

Q8: Design a flowchart for checking whether the number is positive or negative according to the number entered by the user.

Practical No 6

Student Name and Roll Number: Aryan Yadav (24bca004)
Semester /Section: First Sem.
Link to Code:
Date: 25/10/2024
Faculty Signature:

Objective: To study different operators in c programming language.

Program Outcome: n operator is a symbol that tells the compiler to perform specific mathematical or logical functions. C language is rich in built-in operators and provides the following types of operators –

- Arithmetic Operators
- Relational Operators
- Logical Operators
- Bitwise Operators
- Assignment Operators

Background Study: Operators are the foundation of any programming language. We can define operators as symbols that help us to perform specific mathematical and logical computations on operands. In other words, we can say that an operator operates the operands. For example, '+' is an operator used for addition, as shown below:

`c = a + b;`

Here, '+' is the operator known as the addition operator and 'a' and 'b' are operands. The addition operator tells the compiler to add both of the operands 'a' and 'b'.

Operators in C

	Operator	Type
Unary operator	+ +, - -	Unary operator
Binary operator	+, -, *, /, %	Arithmetic operator
	<, <=, >, >=, ==, !=	Relational operator
	&&, , !	Logical operator
	&, , <<, >>, ~, ^	Bitwise operator
Ternary operator	=, +=, -=, *=, /=, %=	Assignment operator
	?:	Ternary or conditional operator

An operator is a symbol that operates on a value or a variable. For example: + is an operator to perform addition.

C has a wide range of operators to perform various operations.

C Arithmetic Operators

An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).

Operator	Meaning of Operator
+	Addition or unary plus
-	subtraction or unary minus
*	Multiplication
/	Division
%	remainder after division (modulo division)

C Increment and Decrement Operators

C programming has two operators increment ++ and decrement -- to change the value of an operand (constant or variable) by 1.

Increment ++ increases the value by 1 whereas decrement -- decreases the value by 1. These two operators are unary operators, meaning they only operate on a single operand.

C Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

Relational operators are used in decision making and loops.

Operator	Meaning of Operator	Example
==	Equal to	5 == 3 is evaluated to 0
>	Greater than	5 > 3 is evaluated to 1
<	Less than	5 < 3 is evaluated to 0
!=	Not equal to	5 != 3 is evaluated to 1
>=	Greater than or equal to	5 >= 3 is evaluated to 1
<=	Less than or equal to	5 <= 3 is evaluated to 0

C Logical Operators

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making in C programming.

Operator	Meaning	Example
&&	Logical AND. True only if all operands are true	If c = 5 and d = 2 then, expression ((c==5) && (d>5)) equals to 0.
	Logical OR. True only if either one operand is true	If c = 5 and d = 2 then, expression ((c==5) (d>5)) equals to 1.
!	Logical NOT. True only if the operand is 0	If c = 5 then, expression !(c==5) equals to 0.

List of Programs

Q1. Write a program to show arithmetic operators (+, -, *, /, and %)

Code:

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

```
int main() {
    float num1, num2;

    printf("Enter two numbers: ");
    scanf("%f %f", &num1, &num2);

    printf("Addition: %.2f + %.2f = %.2f\n", num1, num2, num1 + num2);
    printf("Subtraction: %.2f - %.2f = %.2f\n", num1, num2, num1 - num2);
    printf("Multiplication: %.2f * %.2f = %.2f\n", num1, num2, num1 * num2);

    if (num2 != 0) {
        printf("Division: %.2f / %.2f = %.2f\n", num1, num2, num1 / num2);
        printf("Modulus: %.0f %% %.0f = %.0f\n", num1, num2, (int)num1 %
(int)num2);
    } else {
        printf("Division by zero is not allowed.\n");
        printf("Modulus by zero is not allowed.\n");
    }

    return 0;
}
```

Output and screenshot:

Output[Clear](#)

```
/tmp/9bRCxgZjmI.o
Enter two numbers: 625 25
Addition: 625.00 + 25.00 = 650.00
Subtraction: 625.00 - 25.00 = 600.00
Multiplication: 625.00 * 25.00 = 15625.00
Division: 625.00 / 25.00 = 25.00
Modulus: 625 % 25 = 0
```

```
=== Code Execution Successful ===
```

Q2. Write a program that performs increment and decrement operators (Pre increment, Post-increment, Pre-Decrement, Post-decrement)

Code:

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

```
int main() {
    int num;
```

```
    printf("Enter an integer: ");
    scanf("%d", &num);
```

```
    printf("Original number: %d\n", num);
```

```
    printf("Pre-increment: %d\n", ++num);
    printf("Post-increment: %d\n", num++); // Returns the value, then increments
    num
```

```
printf("Value after post-increment: %d\n", num); // Show the value after post-  
increment
```

```
    // Pre-decrement  
    printf("Pre-decrement: %d\n", --num); // Decrements num and then returns  
the value  
    // Post-decrement  
    printf("Post-decrement: %d\n", num--); // Returns the value, then decrements  
num  
    printf("Value after post-decrement: %d\n", num); // Show the value after  
post-decrement
```

```
    return 0;  
}
```

Output and screenshot:

Output Clear

```
/tmp/qPMmZs9tM1.o  
Enter an integer: 20  
Original number: 20  
Pre-increment: 21  
Post-increment: 21  
Value after post-increment: 22  
Pre-decrement: 21  
Post-decrement: 21  
Value after post-decrement: 20  
  
=== Code Execution Successful ===
```


Q3. Write a program that performs relational operators

Code:

```
#include <stdio.h>

int main() {
    int num1, num2;

    // Input two numbers from the user
    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    // Perform relational operations and display results
    printf("%d > %d: %s\n", num1, num2, (num1 > num2) ? "true" : "false"); //
    Greater than
    printf("%d < %d: %s\n", num1, num2, (num1 < num2) ? "true" : "false"); //
    Less than
    printf("%d >= %d: %s\n", num1, num2, (num1 >= num2) ? "true" : "false");
    // Greater than or equal to
    printf("%d <= %d: %s\n", num1, num2, (num1 <= num2) ? "true" : "false");
    // Less than or equal to
    printf("%d == %d: %s\n", num1, num2, (num1 == num2) ? "true" : "false");
    // Equal to
    printf("%d != %d: %s\n", num1, num2, (num1 != num2) ? "true" : "false"); //
    Not equal to

    return 0;
}
```

Output and screenshot:

Output Clear

```
/tmp/ogqiksfdwB.o
Enter two integers: 20 65
20 > 65: false
20 < 65: true
20 >= 65: false
20 <= 65: true
20 == 65: false
20 != 65: true

=== Code Execution Successful ===
```

Q5. Write a program that performs logical operators

Code:

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

```
int main() {
    int num1, num2;

    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);

    if (num1 > 0 && num2 > 0) {
        printf("Both numbers are positive.\n");
    } else if (num1 < 0 && num2 < 0) {
        printf("Both numbers are negative.\n");
    } else if (num1 > 0 || num2 > 0) {
        printf("At least one number is positive.\n");
    } else {
        printf("Both numbers are zero or negative.\n");
    }
}
```

```
if (!(num1 < 0)) {  
    printf("%d is non-negative.\n", num1);  
} else {  
    printf("%d is negative.\n", num1);  
}  
  
if (!(num2 < 0)) {  
    printf("%d is non-negative.\n", num2);  
} else {  
    printf("%d is negative.\n", num2);  
}  
  
return 0;  
}
```

Output and screenshot:

Output Clear

```
/tmp/qsKe6g4NKm.o  
Enter two integers: 10 -50  
At least one number is positive.  
10 is non-negative.  
-50 is negative.  
  
=== Code Execution Successful ===
```

PRACTICAL NO. 7

Student Name and Roll Number: Aryan Yadav (24bca004)
Semester /Section: First Sem.
Link to Code:
Date: 25/10/2024
Faculty Signature:

	Objective To familiarize the students with operators in C										
	Program Outcome Through this practical, students will learn the concept of operators in C programming										
	Background Study: An operator is a symbol that operates on a value or a variable. For example: + is an operator to perform addition. C has a wide range of operators to perform various operations. <i>C Arithmetic Operators</i> An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).										
	<table> <tr> <th>Operator</th><th>Meaning of Operator</th></tr> <tr> <td>+</td><td>Addition or unary plus</td></tr> <tr> <td>-</td><td>subtraction or unary minus</td></tr> <tr> <td>*</td><td>Multiplication</td></tr> <tr> <td>/</td><td>Division</td></tr> </table>	Operator	Meaning of Operator	+	Addition or unary plus	-	subtraction or unary minus	*	Multiplication	/	Division
Operator	Meaning of Operator										
+	Addition or unary plus										
-	subtraction or unary minus										
*	Multiplication										
/	Division										

%	remainder after division (modulo division)
---	--

C Increment and Decrement Operators

C programming has two operators increment ++ and decrement -- to change the value of an operand (constant or variable) by 1.

Increment ++ increases the value by 1 whereas decrement -- decreases the value by 1. These two operators are unary operators, meaning they only operate on a single operand.

C Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

Relational operators are used in decision making and loops.

Operator	Meaning of Operator	Example
==	Equal to	5 == 3 is evaluated to 0
>	Greater than	5 > 3 is evaluated to 1
<	Less than	5 < 3 is evaluated to 0
!=	Not equal to	5 != 3 is evaluated to 1
>=	Greater than or equal to	5 >= 3 is evaluated to 1
<=	Less than or equal to	5 <= 3 is evaluated to 0

C Logical Operators

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making in C programming.

Operator	Meaning	Example
----------	---------	---------

&&	Logical AND. True only if all operands are true	If $c = 5$ and $d = 2$ then, expression $((c==5) \ \&\& \ (d>5))$ equals to 0.
	Logical OR. True only if either one operand is true	If $c = 5$ and $d = 2$ then, expression $((c==5) \ \ (d>5))$ equals to 1.
!	Logical NOT. True only if the operand is 0	If $c = 5$ then, expression $!(c==5)$ equals to 0.

Q1. What is the output of the following code snippet?

```
#include<stdio.h>

main()
{
    int x = 5;

    if(x==5)
    {
        if(x==5) break;
        printf("Hello");
    }
    printf("Hi");
}
```

Q2. What is the output of the following code snippet?

```
#include<stdio.h>

main()
{
    const int a = 5;

    a++;
    printf("%d", a);
}
```

Q3. What is the output of the following code snippet?

```
#include<stdio.h>

main()
{
    char c = 'A'+255;

    printf("%c", c);
}
```

Q4. What is the output of the below code snippet?

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

```
main()
{
    int a = 1;
    float b = 1.3;
    double c;

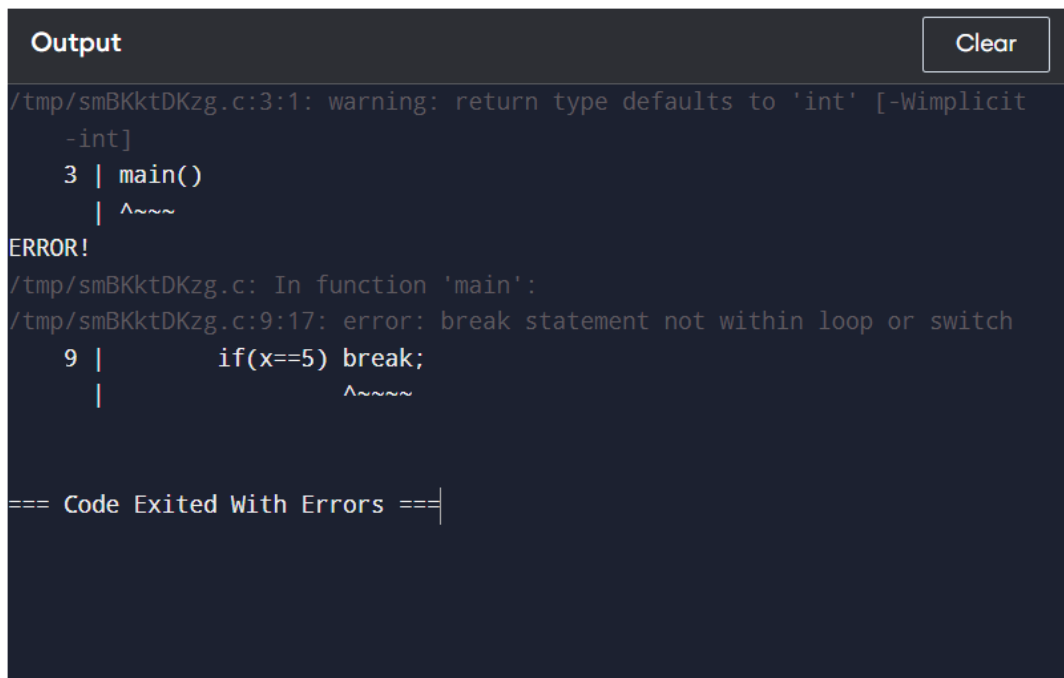
    c = a + b;
    printf("%.2lf", c);
}
```

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Output 1: This code will give error.

Screenshot:



```
Output Clear
/tmp/smBKktDKzg.c:3:1: warning: return type defaults to 'int' [-Wimplicit
-int]
  3 | main()
    | ^~~~
ERROR!
/tmp/smBKktDKzg.c: In function 'main':
/tmp/smBKktDKzg.c:9:17: error: break statement not within loop or switch
  9 |         if(x==5) break;
    |                 ^~~~~
=== Code Exited With Errors ===
```

Output 2: This code will give an error due to an attempt to modify a constant variable.

Screenshot:

```
Output Clear
/tmp/9KmwehRQiU.c:3:1: warning: return type defaults to 'int' [-Wimplicit
-int]
  3 | main()
    | ^~~~
ERROR!
/tmp/9KmwehRQiU.c: In function 'main':
/tmp/9KmwehRQiU.c:7:5: error: increment of read-only variable 'a'
  7 |     a++;
    |     ^~

=== Code Exited With Errors ===
```

Output 3: This code will also give an error due to overflow.

Screenshot:

Output**Clear**

```
/tmp/fyqKv0JN3D.c:3:1: warning: return type defaults to 'int' [-Wimplicit
-int]
3 | main()
  | ^~~~
/tmp/fyqKv0JN3D.c: In function 'main':
/tmp/fyqKv0JN3D.c:5:14: warning: overflow in conversion from 'int' to 'char'
changes value from '320' to '64' [-Woverflow]
5 |     char c = 'A'+255;
  |               ^~~
/tmp/fyqKv0JN3D.o
@

=== Code Execution Successful ===
```

Output 4: Output of the code will be: 2.30

Screenshot:

Output**Clear**

```
/tmp/Y70hKThh5y.c:3:1: warning: return type defaults to 'int' [-Wimplicit  
-int]
```

```
3 | main()  
  | ^~~~
```

```
/tmp/Y70hKThh5y.o
```

```
2.30
```

```
=== Code Execution Successful ===
```

Experiment- 8

Student Name and Roll Number: Aryan Yadav (24bca004)

Semester /Section: First Sem.

Link to Code:

Date: 25/10/2024

Faculty Signature:

Objective

To familiarize the students with if-else loop.

Program Outcome

The students will learn the concept of looping in C. They will be able to understand the different types of statements encountered in C.

Problem Statement

1. While purchasing certain items, a discount of 10% is offered if the quantity purchased is more than 1000. If quantity and price per item are input through the keyboard, write a program to calculate the total expenses.
2. The marks obtained by a student in 5 different subjects are input through the keyboard. The student gets a division as per the following rules: Percentage above or equal to 60 - First division Percentage between 50 and 59 - Second division Percentage between 40 and 49 - Third division Percentage less than 40 - Fail Write a program to calculate the division obtained by the student.
3. Write a program to check whether a triangle is valid or not, when the three angles of the triangle are entered through the keyboard. A triangle is valid if the sum of all the three angles is equal to 180 degrees.

4. Write a program in C to read the age and display whether the candidate is eligible to vote or not.

Definition of Done:

- The program should ask the user to enter a value depicting his age.
- The program should use if-else statement

Background

The syntax of the if statement in C programming is:

```
if (test expression)
```

```
{
```

```
    // statements to be executed if the test expression is true
```

```
}
```

How if statement works?

The if statement evaluates the test expression inside the parenthesis ().

1. If the test expression is evaluated to true, statements inside the body of if are executed.
2. If the test expression is evaluated to false, statements inside the body of if are not executed.

Flipped Questions

1. What is the usage of nested if statements?

2. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
    int i = 5, j = 6, k = 7;
    if(i > j == k)
        printf("%d %d %d", i++, ++j, --k);
    else
        printf("%d %d %d", i, j, k);
    return 0;
}
```

3. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
    int i = 2;
    if(i == (1, 2))
        printf("Hai");
    else
        printf("No Hai");
    return 0;
}
```

a) Compilation error
b) Runtime error
c) Hai
d) No Hai

4. What will be the output of the C program?

```
#include<stdio.h>
int main()
```

```
{  
    if(sizeof(0))  
        printf("Hai");  
    else  
        printf("Bye");  
    return 0;  
}  
a) Hai  
b) Bye  
c) Compilation Error  
d) None
```

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Solution of Problem Statements:

1. #include <stdio.h>

```
float calculate_total_expenses(int quantity, float price_per_item) {  
    float total_price = quantity * price_per_item;  
  
    // Apply 10% discount if quantity is more than 1000  
    if (quantity > 1000) {  
        total_price *= 0.9;  
    }  
}
```

```
    return total_price;
}

int main() {
    int quantity;
    float price_per_item, total_expenses;

    // Input quantity and price per item
    printf("Enter the quantity purchased: ");
    scanf("%d", &quantity);

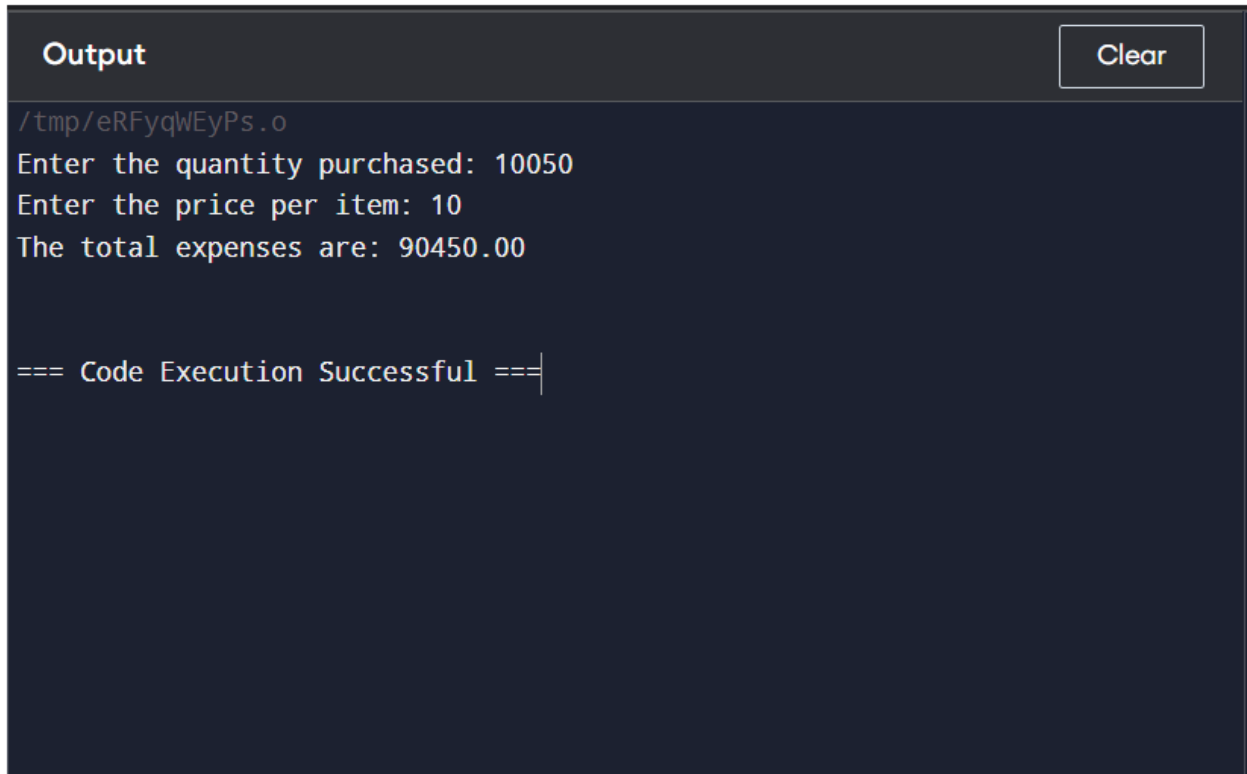
    printf("Enter the price per item: ");
    scanf("%f", &price_per_item);

    // Calculate total expenses
    total_expenses = calculate_total_expenses(quantity, price_per_item);

    // Print the total expenses
    printf("The total expenses are: %.2f\n", total_expenses);

    return 0;
}
```


Screenshot:



The screenshot shows a dark-themed output window titled "Output" with a "Clear" button in the top right corner. The text inside the window displays the execution of a program. It starts with a file path, followed by prompts for quantity and price, and then the calculated total expenses. The execution concludes with a success message.

```
Output Clear  
/tmp/eRFyqWEyPs.o  
Enter the quantity purchased: 10050  
Enter the price per item: 10  
The total expenses are: 90450.00  
  
=== Code Execution Successful ===
```

2. #include <stdio.h>

```
int main() {
```

```
    float mark1, mark2, mark3, mark4, mark5, total, percentage;
```

```
    // Input marks for 5 subjects
```

```
    printf("Enter the marks obtained in 5 subjects:\n");
```

```
    printf("Subject 1: ");
```

```
    scanf("%f", &mark1);
```

```
printf("Subject 2: ");  
scanf("%f", &mark2);
```

```
printf("Subject 3: ");  
scanf("%f", &mark3);
```

```
printf("Subject 4: ");  
scanf("%f", &mark4);
```

```
printf("Subject 5: ");  
scanf("%f", &mark5);
```

```
// Calculate total and percentage
```

```
total = mark1 + mark2 + mark3 + mark4 + mark5;
```

```
percentage = total / 5;
```

```
// Print the percentage and the division obtained
```

```
printf("The percentage of the student is: %.2f\n", percentage);
```

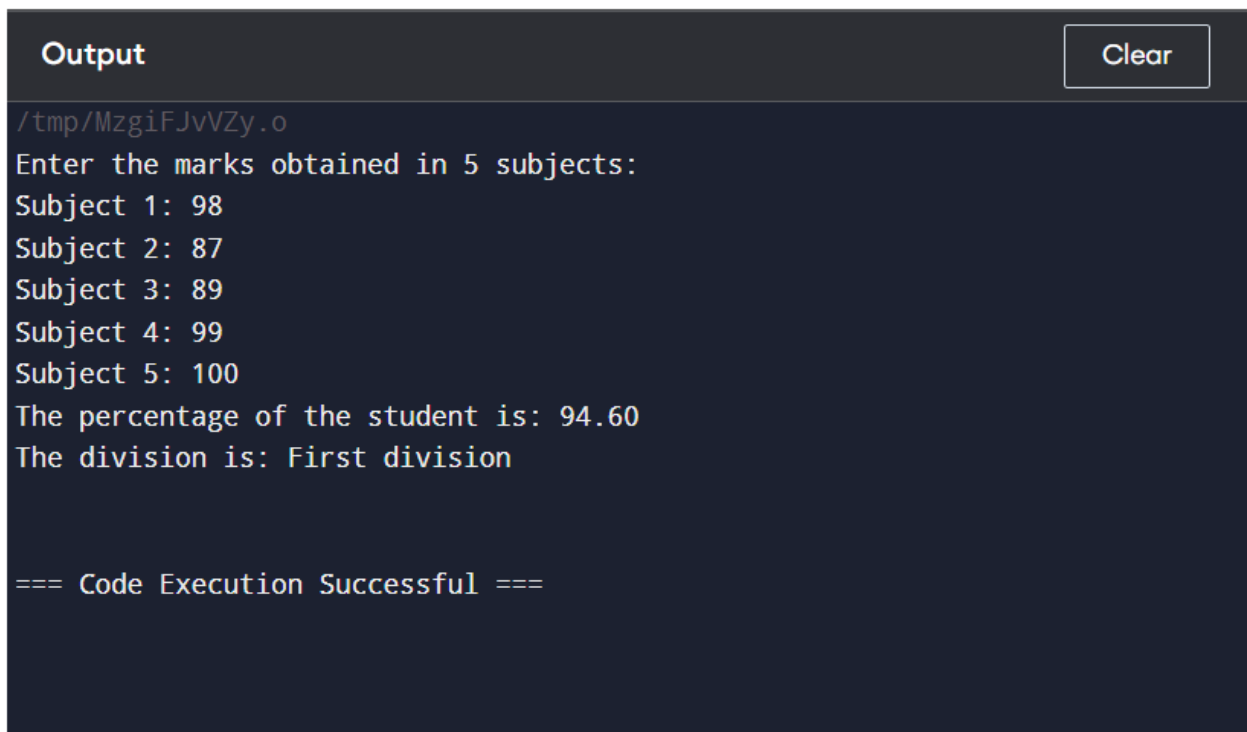
```
if (percentage >= 60) {
```

```
    printf("The division is: First division\n");
```

```
} else if (percentage >= 50) {
```

```
printf("The division is: Second division\n");  
} else if (percentage >= 40) {  
    printf("The division is: Third division\n");  
} else {  
    printf("The division is: Fail\n");  
}  
  
return 0;  
}
```

Screenshot:



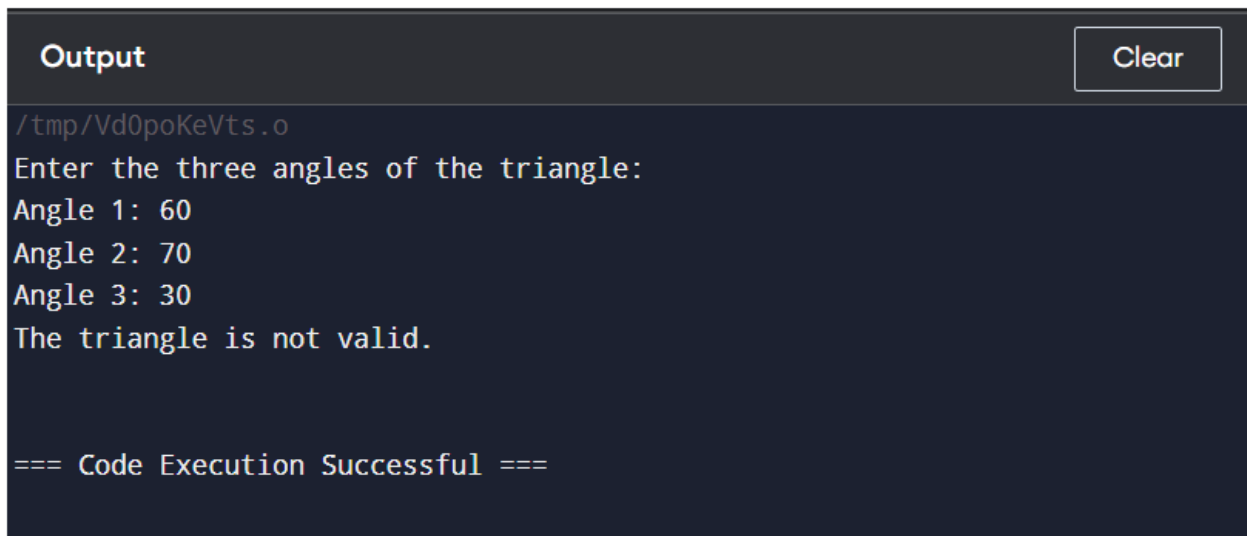
```
Output Clear  
/tmp/MzgiFJvVZy.o  
Enter the marks obtained in 5 subjects:  
Subject 1: 98  
Subject 2: 87  
Subject 3: 89  
Subject 4: 99  
Subject 5: 100  
The percentage of the student is: 94.60  
The division is: First division  
  
=== Code Execution Successful ===
```

3. #include <stdio.h>

```
int main() {  
    int angle1, angle2, angle3, sum;  
  
    // Input the three angles of the triangle  
    printf("Enter the three angles of the triangle:\n");  
  
    printf("Angle 1: ");  
    scanf("%d", &angle1);  
  
    printf("Angle 2: ");  
    scanf("%d", &angle2);  
  
    printf("Angle 3: ");  
    scanf("%d", &angle3);  
  
    // Calculate the sum of the angles  
    sum = angle1 + angle2 + angle3;  
  
    // Check if the triangle is valid  
    if (sum == 180) {  
        printf("The triangle is valid.\n");  
    } else {
```

```
        printf("The triangle is not valid.\n");  
    }  
  
    return 0;  
}
```

Screenshot:



The screenshot shows a dark-themed terminal window titled "Output" with a "Clear" button in the top right corner. The terminal content is as follows:

```
/tmp/Vd0poKeVts.o  
Enter the three angles of the triangle:  
Angle 1: 60  
Angle 2: 70  
Angle 3: 30  
The triangle is not valid.  
  
=== Code Execution Successful ===
```

4. #include <stdio.h>

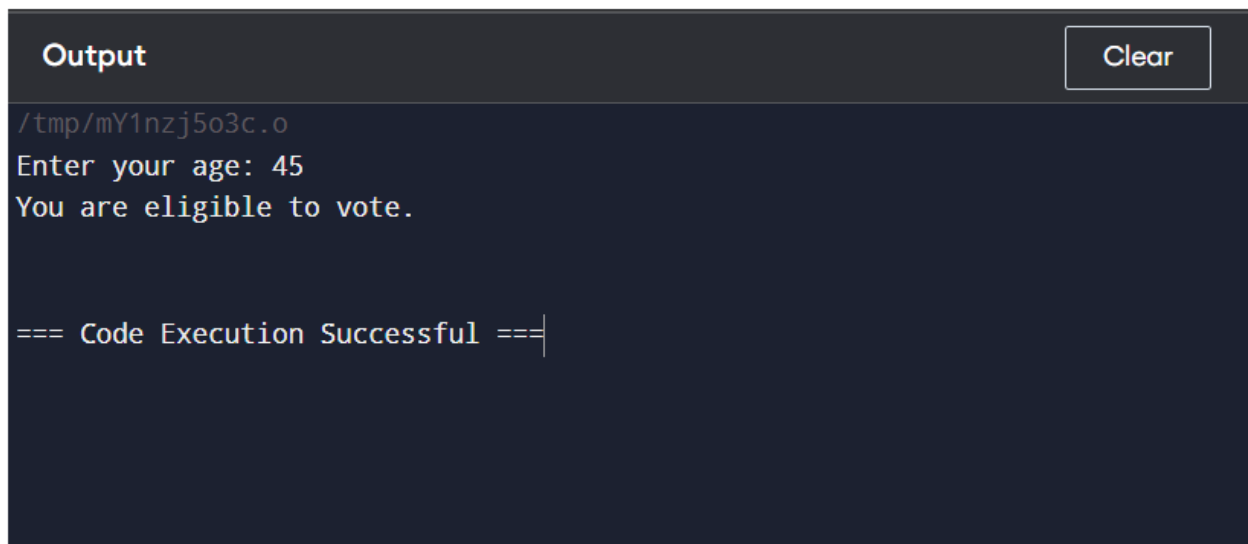
```
int main() {  
    int age;  
  
    // Ask the user to enter their age  
    printf("Enter your age: ");  
    scanf("%d", &age);
```

```
// Check if the candidate is eligible to vote

if (age >= 18) {
    printf("You are eligible to vote.\n");
} else {
    printf("You are not eligible to vote.\n");
}

return 0;
}
```

Screenshot:



The screenshot shows a dark-themed code editor with a terminal window. The terminal output is as follows:

```
/tmp/mY1nzj5o3c.o
Enter your age: 45
You are eligible to vote.

=== Code Execution Successful ===
```

There is a 'Clear' button in the top right corner of the terminal window.

Solution to Questions:

1. Usage of Nested If Statements

Nested if statements are used to check multiple conditions where one condition relies on another. They help in:

1. Complex Decision-Making: To evaluate several related conditions.
2. Specific Cases: To handle different scenarios based on combined criteria.

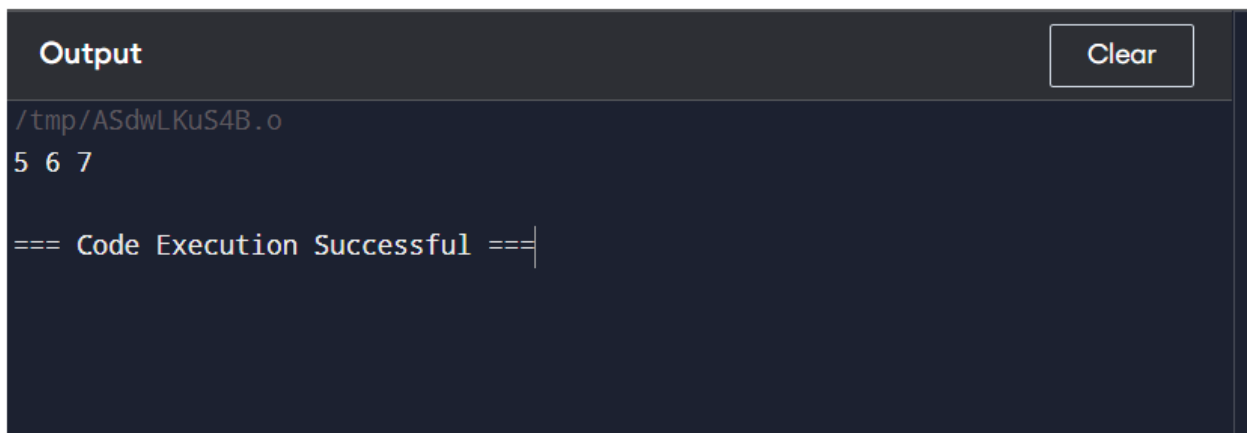
Example

```
if (age >= 18) {  
    if (citizen) {  
        printf("You can vote.");  
    }  
}
```

In this example, it checks if a person is old enough and if they are a citizen before allowing them to vote.

2. Output: 5 6 7

Screenshot:



The screenshot shows a dark-themed window titled "Output" with a "Clear" button in the top right corner. The output text is as follows:

```
/tmp/ASdwLKuS4B.o  
5 6 7  
  
=== Code Execution Successful ===
```

3. Output: c) Hai Screenshot:

```
Output Clear  
/tmp/4SdU52ihA9.o  
Hai  
  
=== Code Execution Successful ===
```

4. Output: a) Hai

Screenshot:

```
Output Clear  
/tmp/hY0hkbhZfg.o  
Hai  
  
=== Code Execution Successful ===
```


Experiment- 9

Student Name and Roll Number: Aryan Yadav (24bca004)

Semester /Section: First Sem.

Link to Code:

Date: 25/10/2024

Faculty Signature:

Objective

To familiarize the students with while and do while loop along with jumping statement.

Program Outcome

The students will be able to understand iterative control statements and how to apply while and do-while loop in programs.

Problem Statement

1. Find factorial of a number using while loop
2. Write a program in C to print the table of a given number using do while loop.
3. Write a program to check whether a number is prime or not.

Background***while loop***

The syntax of the while loop is:

```
while (testExpression)
{
    // statements inside the body of the loop
}
```

How while loop works?

- The while loop evaluates the test expression inside the parenthesis ().
- If the test expression is true, statements inside the body of while loop are executed. Then, the test expression is evaluated again.
- The process goes on until the test expression is evaluated to false.
- If the test expression is false, the loop terminates (ends).

do...while loop

The do..while loop is similar to the while loop with one important difference. The body of do...while loop is executed at least once. Only then, the test expression is evaluated.

The syntax of the do...while loop is:

```
do
{
    // statements inside the body of the loop
}
```

```
while (testExpression);
```

How do...while loop works?

- The body of do-while loop is executed once. Only then, the test expression is evaluated.
- If the test expression is true, the body of the loop is executed again and the test expression is evaluated.
- This process goes on until the test expression becomes false.

- If the test expression is false, the loop ends.

Question Bank

Suggested Question Bank

1. Differentiate between while and do while loop.

2. What will be the output of the C program?

a) void main()

```
{  
    int j =1;  
    while ( j <= 10 )  
    {  
        printf ( "\n%d", j ) ;  
        j = j + 1 ; } }
```

b) void main()

```
{  
    float x = 1.1 ;  
    while ( x == 1.1 )  
    {  
        printf ( "\n%f", x ) ;  
        x = x - 0.1 ; } }
```

Flipped Questions**1. What will be the output of the following?**

a) #include <stdio.h>

```
int main()
{ int i = 0;
  while (i = 0)
    printf("True\n");
    printf("False\n");
}
```

b) #include<stdio.h>

```
int main()
{ int i = 0;
  while(i < 3, i = 0, i < 5)
  { printf("Loop ");
    i++; }
  return 0; }
```

c) #include<stdio.h>

```
int main()
{ int i = 0;
  while(i++)
  {
    printf("Loop ");
    if(i == 3) break; }
  return 0; }
```

Solution To Problem Statements:

1. Code:

```
#include <stdio.h>

int main() {
    int num;
    long double factorial = 1.0;

    printf("Enter a number: ");
    scanf("%d", &num);

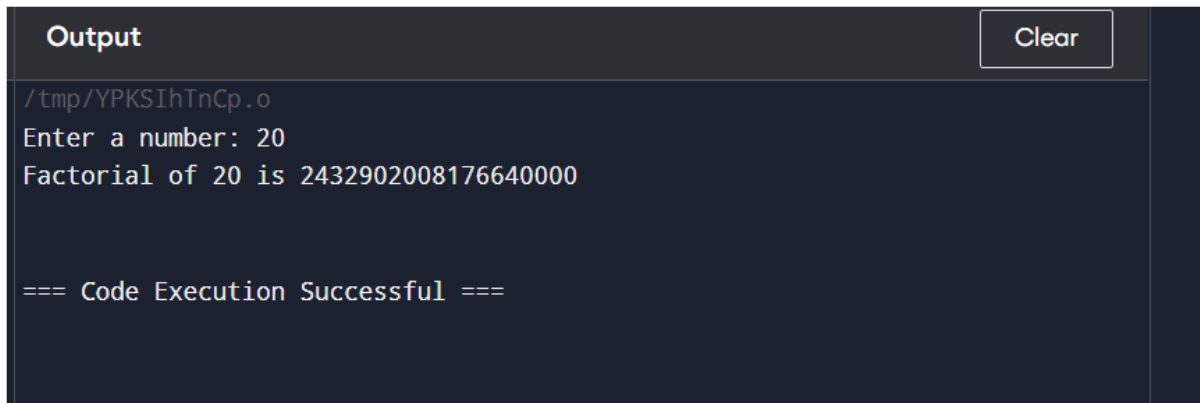
    int i = 1;
    while (i <= num) {
        factorial *= i;
        i++;
    }

    printf("Factorial of %d is %.0Lf\n", num, factorial);

    return 0;
```

```
}
```

Screenshot:



The screenshot shows a dark-themed output window titled "Output" with a "Clear" button. The text inside the window displays the execution of a program that calculates the factorial of a user-input number. The output shows the file path, the input "20", the resulting factorial value, and a success message.

```
/tmp/YPKSIhTnCp.o
Enter a number: 20
Factorial of 20 is 2432902008176640000

=== Code Execution Successful ===
```

2. Code:

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

```
int main() {
```

```
    int number, i = 1;
```

```
    printf("Enter a number: ");
```

```
    scanf("%d", &number);
```

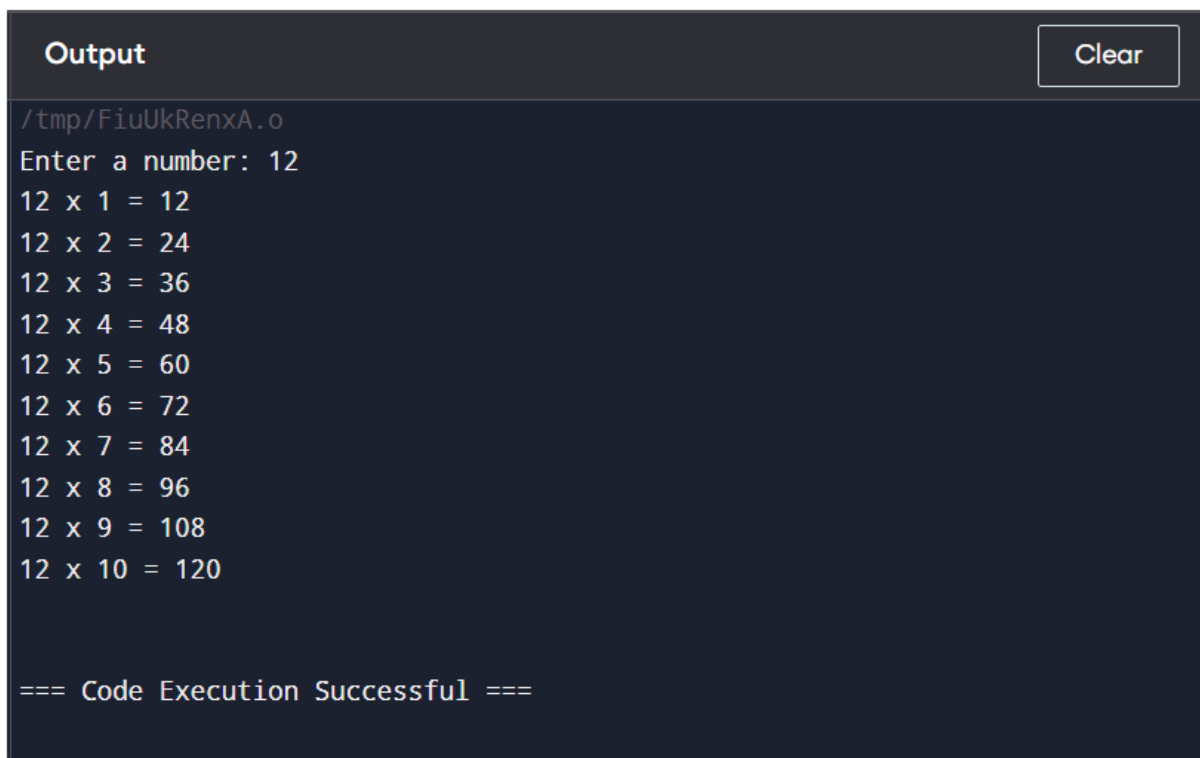
```
    do {
```

```
        printf("%d x %d = %d\n", number, i, number * i);
```

```
        i++;
```

```
    } while (i <= 10);  
  
    return 0;  
}
```

Screenshot:



```
Output  
/tmp/FiuUkRenxA.o  
Enter a number: 12  
12 x 1 = 12  
12 x 2 = 24  
12 x 3 = 36  
12 x 4 = 48  
12 x 5 = 60  
12 x 6 = 72  
12 x 7 = 84  
12 x 8 = 96  
12 x 9 = 108  
12 x 10 = 120  
  
=== Code Execution Successful ===
```

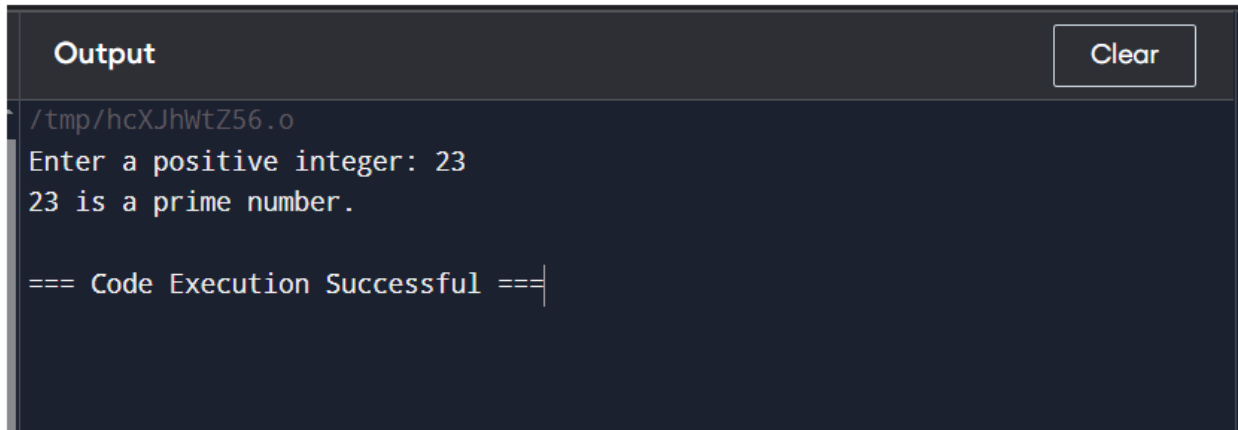
3. Code:

```
#include <stdio.h>  
  
int main()  
{  
    int n, i, flag = 0;
```

```
printf("Enter a positive integer: ");  
scanf("%d", &n);  
for (i = 2; i <= n / 2; ++i)  
{  
    if (n % i == 0)  
    {  
        flag = 1;  
        break;  
    }  
}  
if (n == 1)  
{  
    printf("1 is neither prime nor composite.");  
}  
else  
{  
    if (flag == 0)  
        printf("%d is a prime number.", n);  
    else  
        printf("%d is not a prime number.", n);  
}  
return 0;
```


}

Screenshot:



```
Output Clear
/tmp/hcXJhwtZ56.o
Enter a positive integer: 23
23 is a prime number.

=== Code Execution Successful ===
```

Solution For Question Bank:

1. The main difference between a while loop and a do-while loop is when they check the condition:

- While loop: It checks the condition before running the loop's code. If the condition is false from the start, the code inside the loop won't run at all.

Example:

```
while (condition) {
    // code
}
```

The loop runs only if the condition is true.

- Do-while loop: It runs the code first, and then checks the condition. This means the code inside the loop will always run at least once, even if the condition is false.

Example:

```
do {  
    // code  
} while (condition);
```

The loop runs once, and then checks the condition to decide whether to continue.

2.

a.) Output:

1

2

3

4

5

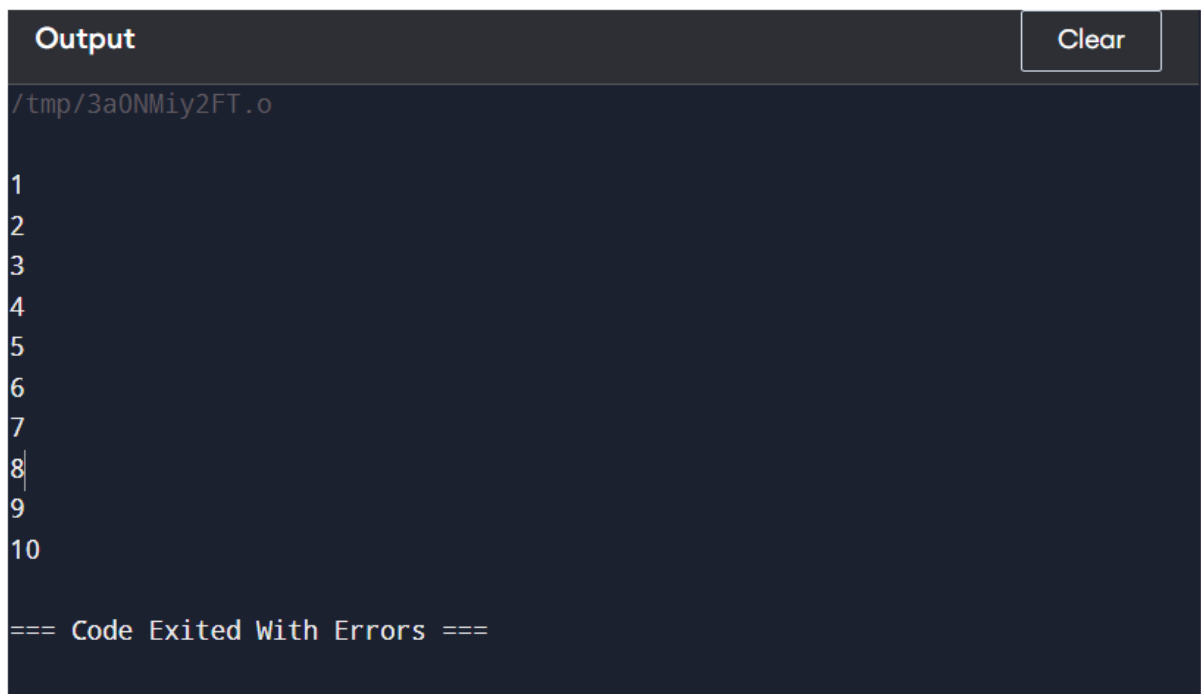
6

7

8

9

10

Screenshot:

```
Output Clear
/tmp/3a0NMiy2FT.o
1
2
3
4
5
6
7
8
9
10

=== Code Exited With Errors ===
```

b). The loop will not run, and nothing will be printed. The program simply terminates without any output.

Solution to flipped Questions:**1. a.) Output : False****Screenshot:**

```
Output Clear  
/tmp/naGdpFJJ9Y.o  
False  
  
=== Code Execution Successful ===
```

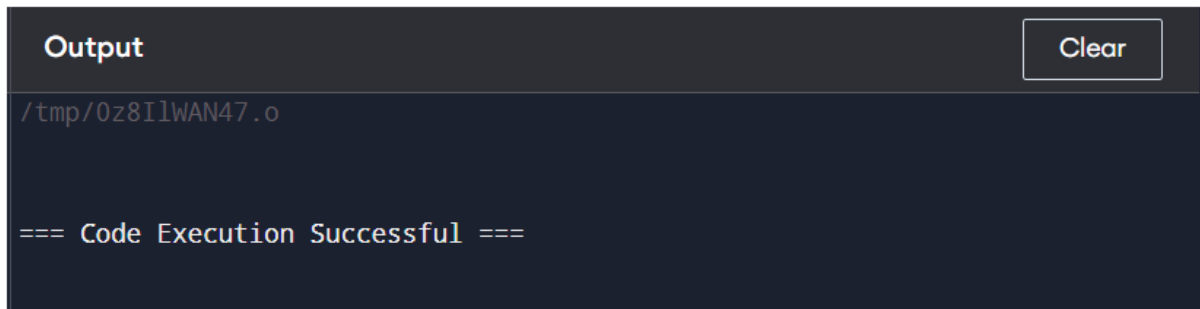
b.) Output: The output will continuously print "Loop " indefinitely because i will never be able to increment properly.

Screenshot:

[illegible]

c.) Output: Because `i` starts at 0, the loop condition `while(i++)` is false on the first check, so nothing is printed. The program terminates without output.

Screenshot:



```
Output Clear  
/tmp/0z8IlWAN47.o  
  
=== Code Execution Successful ===
```

Experiment- 10

Student Name and Roll Number: Aryan Yadav (24bca004)

Semester /Section: First Sem.

Link to Code:

Date: 25/10/2024

Faculty Signature:

Objective

To familiarize the students with for loop

Program Outcome

The students will be able to understand iterative control statements and how to apply for loop in programs.

Problem Statement

Q1. Write a c program to print numbers from 1 to 10

Code :

Q2. Program to calculate the sum of first n natural numbers

// Positive integers 1,2,3...n are known as natural numbers

Code:

3. Write a program to print the following pattern.

1

1 2

1 2 3

1 2 3 4

Background

for Loop

The syntax of the for loop is:

```
for (initializationStatement; testExpression; updateStatement)
```

```
{ // statements inside the body of loop }
```

How for loop works?

- The initialization statement is executed only once.
- Then, the test expression is evaluated. If the test expression is evaluated to false, the for loop is terminated.
- However, if the test expression is evaluated to true, statements inside the body of for loop are executed, and the update expression is updated.
- Again the test expression is evaluated.
- This process goes on until the test expression is false. When the test expression is false, the loop terminates.

Question Bank

1. To find the sum of individual digits of a positive integer and test given number is palindrome.
2. That will read in a positive integer value and determine whether its prime or Fibonacci.

Flipped Questions

1. How many times "FOCP" is get printed?

```
#include<stdio.h>
int main()
{
    int x;
    for(x=-1; x<=10; x++)
    {
        if(x < 5)
            continue;
        else
            break;
        printf("FOCP");
    }
    return 0;
}
```

- 2 What will be the output of the program?

```
#include<stdio.h>
int main()
{
    int i=0;
    for(; i<=5; i++);
    printf("%d", i);
    return 0;
}
```

3. Point out the error, if any in the for loop.

```
#include<stdio.h>
int main()
{
    int i=1;
```



```
for(;;)
{
    printf("%d\n", i++);
    if(i>10)
        break;
}
return 0;
}
```

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

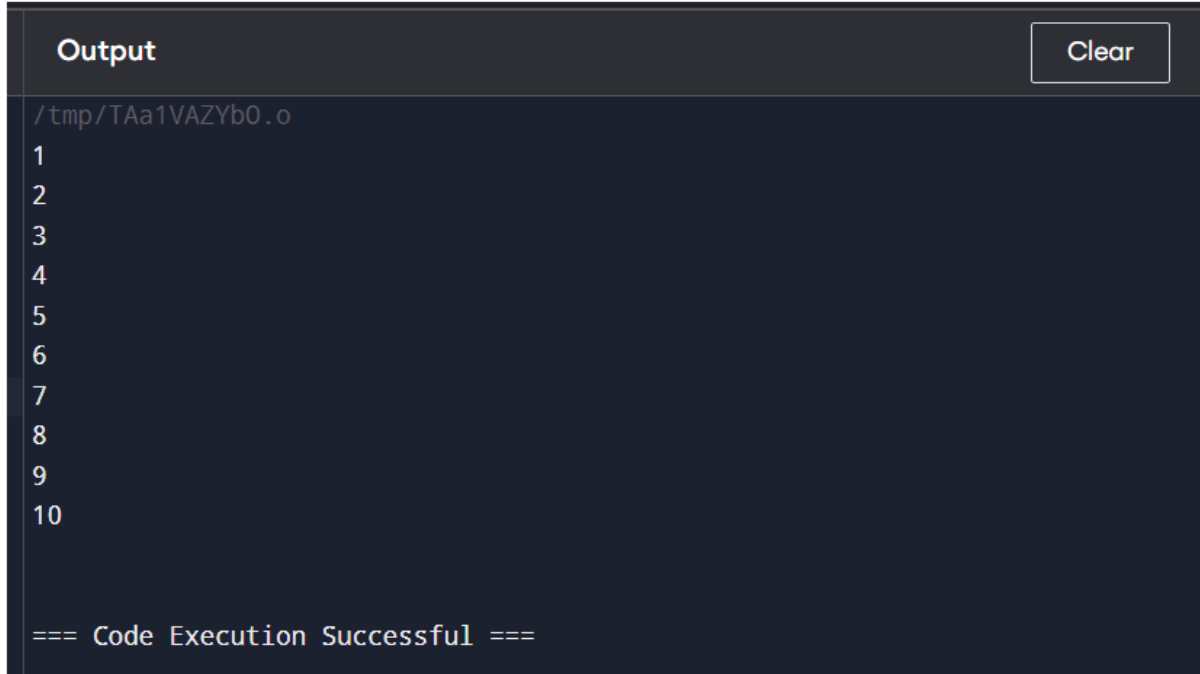
Solutions to Problem Statements:

1. Code:

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

```
int main() {
    for (int i = 1; i <= 10; i++) {
        printf("%d\n", i);
    }
    return 0;
}
```

Screenshot:



The screenshot shows a dark-themed output window titled "Output" with a "Clear" button in the top right corner. The output text is as follows:

```
/tmp/TAa1VAZYb0.o
1
2
3
4
5
6
7
8
9
10

=== Code Execution Successful ===
```

2. Code:

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

```
int main() {
```

```
    int n, sum = 0;
```

```
    printf("Enter a number: ");
```

```
    scanf("%d", &n);
```

```
    for (int i = 1; i <= n; i++) {
```

```
        sum += i;
```

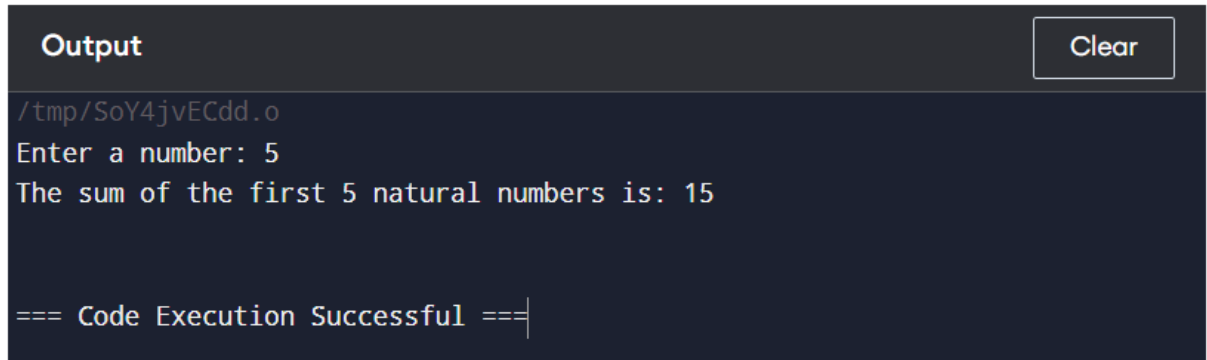
```
    }
```

```
printf("The sum of the first %d natural numbers is: %d\n", n, sum);

return 0;

}
```

Screenshot:



```
Output Clear
/tmp/SoY4jvECdd.o
Enter a number: 5
The sum of the first 5 natural numbers is: 15

=== Code Execution Successful ===
```

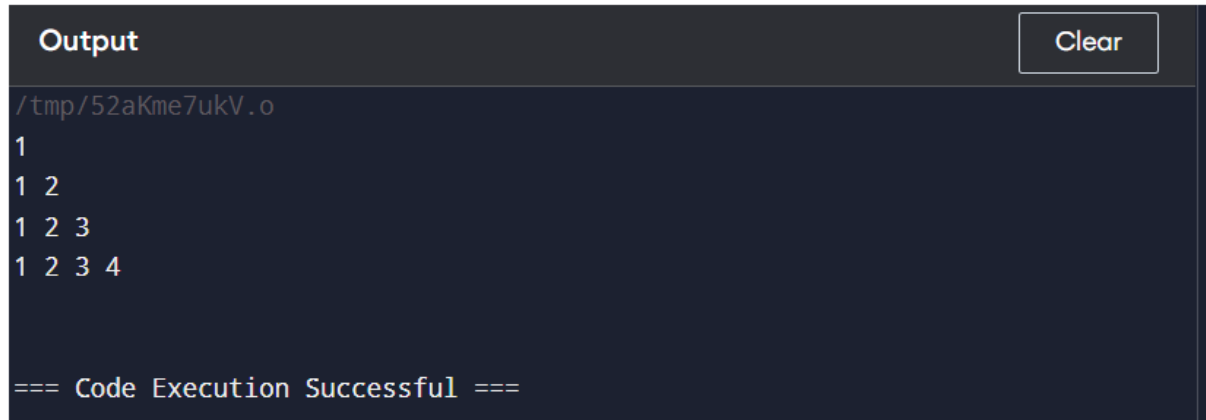
3. Code:

```
#include <stdio.h>

int main() {
    for (int i = 1; i <= 4; i++) {
        for (int j = 1; j <= i; j++) {
            printf("%d ", j);
        }
        printf("\n");
    }
}
```

```
    return 0;  
}
```

Screenshot:



```
Output Clear  
/tmp/52aKme7ukV.o  
1  
1 2  
1 2 3  
1 2 3 4  
  
=== Code Execution Successful ===
```

Solutions to Question Bank

1. Code:

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

```
int main() {
```

```
    int num, originalNum, reversedNum = 0, remainder, sumOfDigits = 0;
```

```
    printf("Enter a positive integer: ");
```

```
    scanf("%d", &num);
```

```
    originalNum = num;
```

```
while(num != 0) {  
    remainder = num % 10;  
    sumOfDigits += remainder;  
    reversedNum = reversedNum * 10 + remainder;  
    num /= 10;  
}  
  
printf("Sum of digits: %d\n", sumOfDigits);  
  
if(originalNum == reversedNum) {  
    printf("The number is a palindrome.\n");  
} else {  
    printf("The number is not a palindrome.\n");  
}  
  
return 0;  
}
```

Screenshot:

```
Output Clear  
/tmp/akTKFWQq1d.o  
Enter a positive integer: 123454321  
Sum of digits: 25  
The number is a palindrome.  
  
=== Code Execution Successful ===
```

2. Code: ____ Unable to form the logic ____

Solutions to flipped Questions.

1. The code does not print "FOCP" at all. Zero times.

2. Output: 6

Screenshot:

```
Output Clear  
/tmp/uneZTYEVAe.o  
6  
  
=== Code Execution Successful ===
```

3. There is no error in the loop. It will print numbers from 1 to 10, each on a new line, and then exit the loop.

Output Screenshot:

Output

Clear

```
/tmp/ZuHmYQJVHz.o
```

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

```
=== Code Execution Successful ===
```