K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent College of Somaiya Vidyavihar University)

Batch: Roll No.: 16010422185
Experiment / assignment / tutorial No. 2
Grade: AA / AB / BB / BC / CC / CD /DD
Signature of the Staff In-charge with date

TITLE: Write a program to accept 3 numbers from the user and find the largest of the 3 numbers using
If - else if-else
Ternary operator

AIM: Write a program to accept 3 numbers from the user and find the largest of the 3 numbers using
If - else if-else
Ternary operator

Expected OUTCOME of Experiment:

Books/ Journals/ Websites referred:

1. Programming in ANSI C, E. Balagurusamy, 7 th Edition, 2016, McGraw-Hill Education, India.
2. Structured Programming Approach, Pradeep Dey and Manas Ghosh, 1 st Edition, 2016, Oxford University Press, India.
3. Let Us C, Yashwant Kanetkar, 15th Edition, 2016, BPB Publications, India.

Problem Definition:

Ask user to input three numbers. Compare three numbers to find the largest of them using
1. Nested if else statement

2. Using ternary operator

Algorithm:

1. Start

2. Read the three numbers to be compared, as number1, number2 and number3. 3. Check if number1 is greater than number2.

3.1 If true, then check if number1 is greater than number3.

If true, print number1 as the greatest number..

If false, print number3 as the greatest number.

3.2 If false, then check if num2 is greater than num3.

If true, print num2 as the greatest number.

If false, print num3 as the greatest number.

4. End

K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent College of Somaiya Vidyavihar University)

Implementation details:

Program using Nested if else statement -

```
#include <stdio.h>

int main()
{

   int num1, num2, num3, max;

   printf("Enter three numbers: ");
   scanf("%d %d %d", &num1, &num2, &num3);

   if (num1 > num2)


    {
          if (num1 > num3)
        {
          max = num1;
        }
      else
        {
         max = num2;
        }
   }
   else
   {
       if (num2 > num3)
        {
          max = num2;
        }
       else
        {
          max = num3;
        }
   }

   printf("%d is largest number.\n", max);

   return 0;
}
```

K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent College of Somaiya Vidyavihar University)

Program using ternary operator -

```
#include <stdio.h>

int main()
{
   int num1, num2, num3, max;

   printf("Enter three numbers: ");
   scanf("%d %d %d", &num1, &num2, &num3);

   max = (num1 > num2) ? (num1 > num3 ? num1 : num3) : (num2 > num3 ? num2
: num3);

   printf("%d is largest number.\n", max);

   return 0;
}
```

Output(s):

Output for Program using Nested if else statement -

```
Enter three numbers: 29
99
109
109 is largest number.
```

Output for Program using ternary operator -

```
Enter three numbers: 66
77
23
77 is largest number.
```

Conclusion:

Largest among 3 numbers is found by 2 methods -

1. Nested if-else statement
2. Ternary operator.

By checking conditions, the largest among 3 numbers is found and the result is

displayed on screen.

K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent College of Somaiya Vidyavihar University)

Post Lab Descriptive Questions

1. **Explain relational, logical and bitwise operators with examples.** 2. **Write associative rules and precedence tables of various operators.**

Ans 1:

1) Relational operators are used to compare two values in C language. It checks the relationship between two values. If the relation is true, it returns 1. However, if the relation is false, it returns 0.

Here is the table of relational operators in C language.

Operators	Operator Name
==	Equal to
>	Greater than
<	Less than
!=	Not equal to
>=	Greater than or equal to

<= Less than or equal to

Example -

```
#include <stdio.h>
int main()
{
   int x = 10, y = 28;
   if (x == y)
      printf("Both variables are equal");
   if (x > y)
      printf("x is greater than y ");
   if (x < y)
      printf("x is less than y ");
   if (x != y)
      printf("x is not equal to y ");
   if (x <= y)
      printf("x is lesser or equal to y");
   if (x >= y)
      printf("x is greater or equal to y ");
   return 0;
}
```

2) Logical operators are used to perform logical operations. It returns 0 or 1 based on the result of the condition, whether it's true or false. These operators are used for decision making in C language.

Here is the table of logical operators in C language.

Operators	**Meaning of Operators Results**
&&	Logical AND True when all operands are true
||	Logical OR True only if either one operand is true
!	Logical NOT True when operand is zero

Department of Science and Humanities

Page No PIC Sem I 2022-23(Odd Sem)

K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent

College of Somaiya Vidyavihar University)

Example -

```
#include <stdio.h>
int main()
{
   int x = 10, y = 28, a = 15, b = 20;
   if (x < y && a == b)
   {
      printf("x is less than y AND a is equal to b");
   }
   if (x < y || a == b)
   {
      printf("x is less than y OR a is equal to b");
   }
   if (!x)
   {
      printf("x is zero");
   }
   return 0;
}
```

3) The Bitwise Operator in C performs its operation on the individual bits of its operand, where operands are values or expressions on which an operator operates. These operators are also used to perform the core actions as well as high-level arithmetic operations that require direct support of the processor.

There are six different Bitwise Operators in C. These are:

1. Bitwise AND operator (&)
2. Bitwise OR operator (|)
3. Bitwise exclusive OR operator (^)
4. Binary One's Complement or Bitwise NOT operator (~)
5. Bitwise Left shift operator (<<)
6. Bitwise Right shift operator (>>)

Example -

x	y	x & y	x | y	x ^ y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Ans 2:

1. The associativity of operators determines the direction in which an expression is evaluated.
2. Also, if two operators of the same precedence (priority) are present, associativity determines the direction in which they execute.
3. Precedence Table -

Precedence	Operator	Operator Meaning	Associativity
1	() [] -> .	Function Call array reference Structure member access Structure member access	Left to Right
2	! ~ + - ++ -- & * sizeof(t yp e)	negation 1's complement Unary Plus Unary minus Increment operator decrement operator address of operator pointer returns size of a variable type conversion	Right to Left
3	* / %	multiplication division remainder	Left to Right
4	+	addition	Left to Right

	-	subtraction	
5	<< >>	left shift right shift	Left to Right
6	< <=	less than less than or equal to	Left to Right
7 8	== != &	equal to not equal to bitwise AND	Left to Right Left to Right
9	^	bitwise EXCLUSIVE OR	Left to Right
10	|	bitwise OR	Left to Right
11 12	&& ||	logical AND logical OR	Left to Right Left to Right
13	?:	Conditional Operator	Left to Right
14	= *= /= %= += -= &= ^= |= <<=	assignment assign multiplication assign division assign remainder assign addition assign subtraction assign bitwise AND assign bitwise XOR assign bitwise OR assign left shift	Right to Left

	>>=	assign right shift	
15	,	separator	Left to Right

Date: _____ Signature of faculty in-charge