K. J. Somaiya College of Engineering, Mumbai-77

(Somaiya Vidyavihar University)

Name: Chandana Ramesh Galgali

Batch: P3-3 Roll No.: 16010422234

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: The user will enter 3 numbers and the largest of the three numbers will be printed on the screen as the output.

Books/ Journals/ Websites referred:

1. Programming in C, second edition, Pradeep Dey and Manas Ghosh, Oxford University Press.
2. Programming in ANSI C, fifth edition, E Balagurusamy, Tata McGraw Hill. 3. Introduction to programming and problem solving , G. Michael Schneider ,Wiley India edition.
4. **http://cse.iitkgp.ac.in/~rkumar/pds-vlab/**

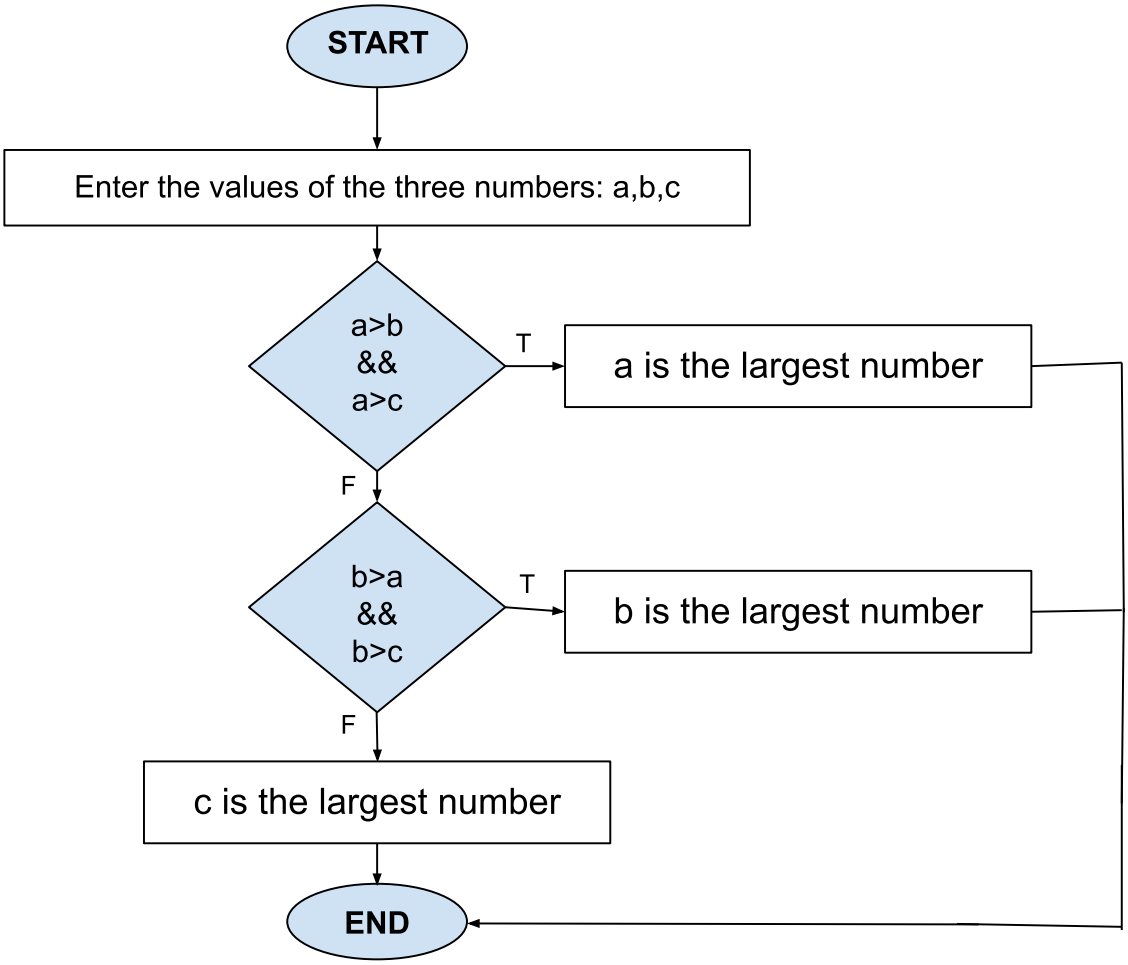
Problem Definition:

Ask the user to input three numbers. Compare the three numbers to find the largest of them using:

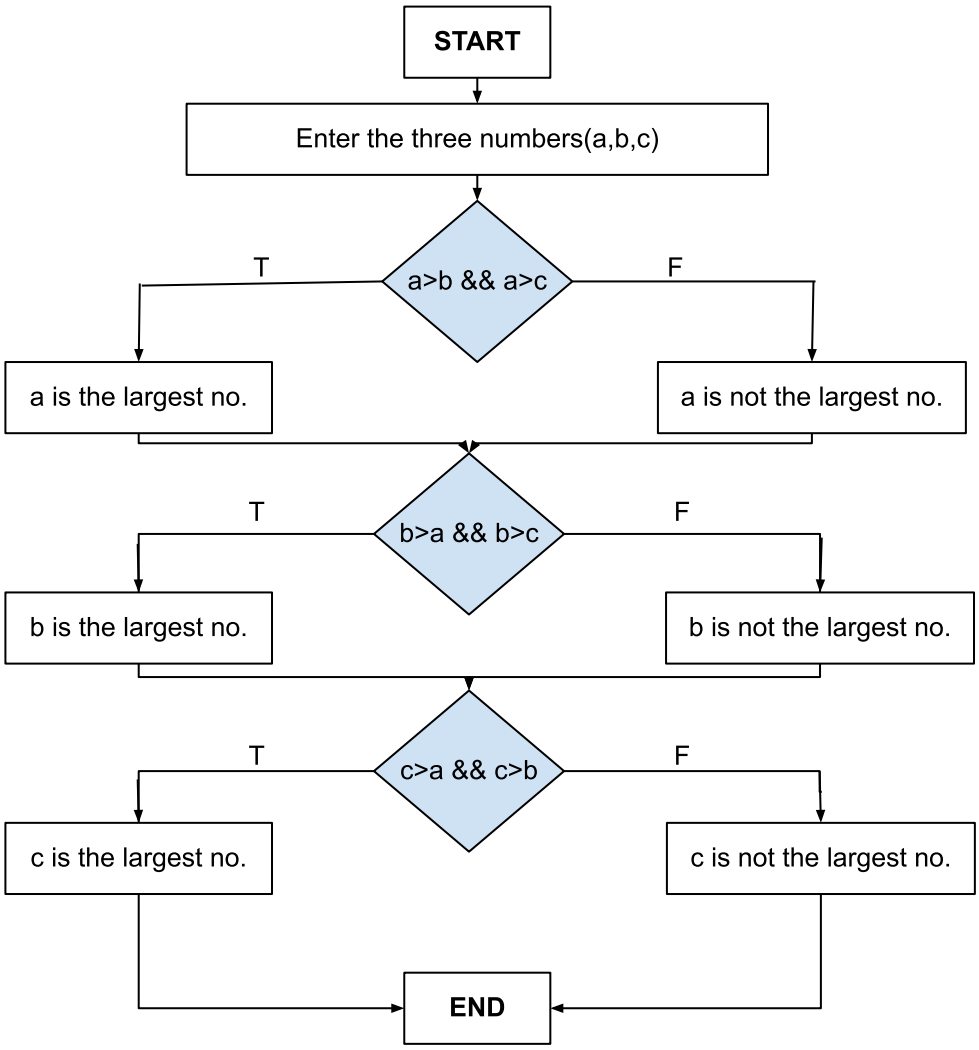
1. Nested if-else statement
2. Using ternary operator

Flowchart:

1.If-else if-else:



2. Ternary Operator:



Implementation details:

1.If-else if-else:

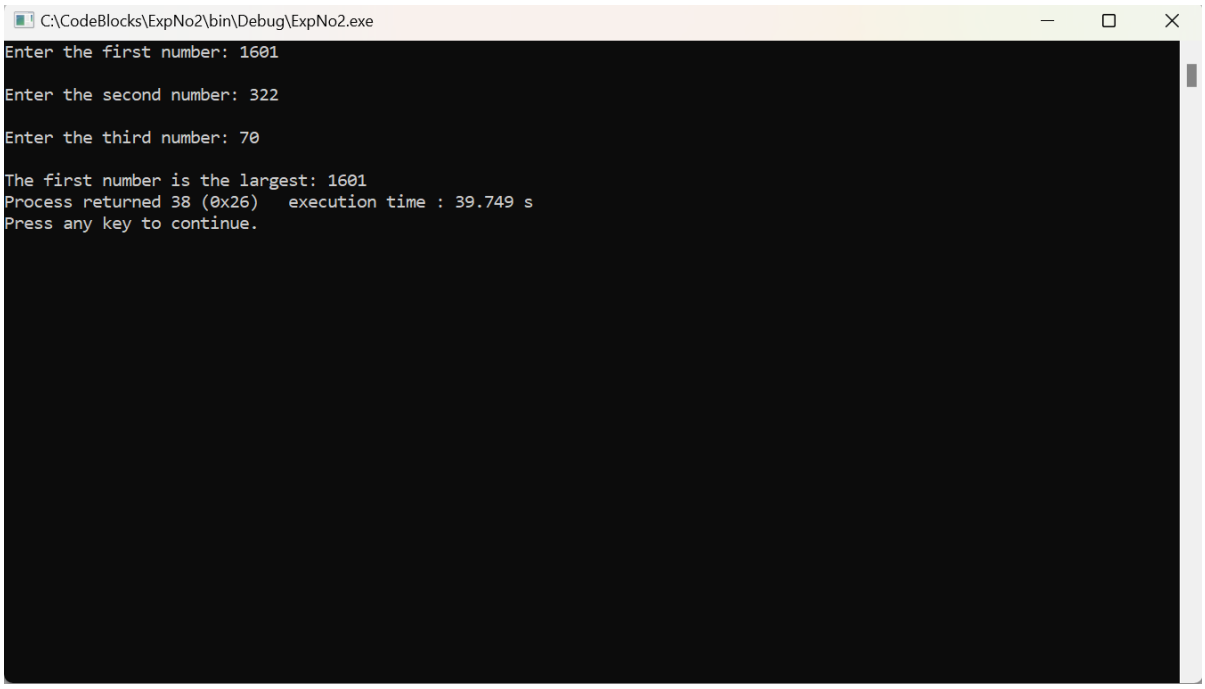
```
#include<stdio.h>
void main()
{
   int a,b,c;
   printf("Enter the first number: ");
   scanf("%d", &a);
   printf("\nEnter the second number: ");
   scanf("%d", &b);
   printf("\nEnter the third number: ");
   scanf("%d", &c);
   if(a>b && a>c)
   {
      printf("\nThe first number is the largest: %d",a);
   }
   else if(b>a && b>c)
   {
      printf("\nThe second number is the largest: %d",b);
   }
   else
   {
      printf("\nThe third number is the largest: %d",c);
   }
}
```

2. Using ternary operator:

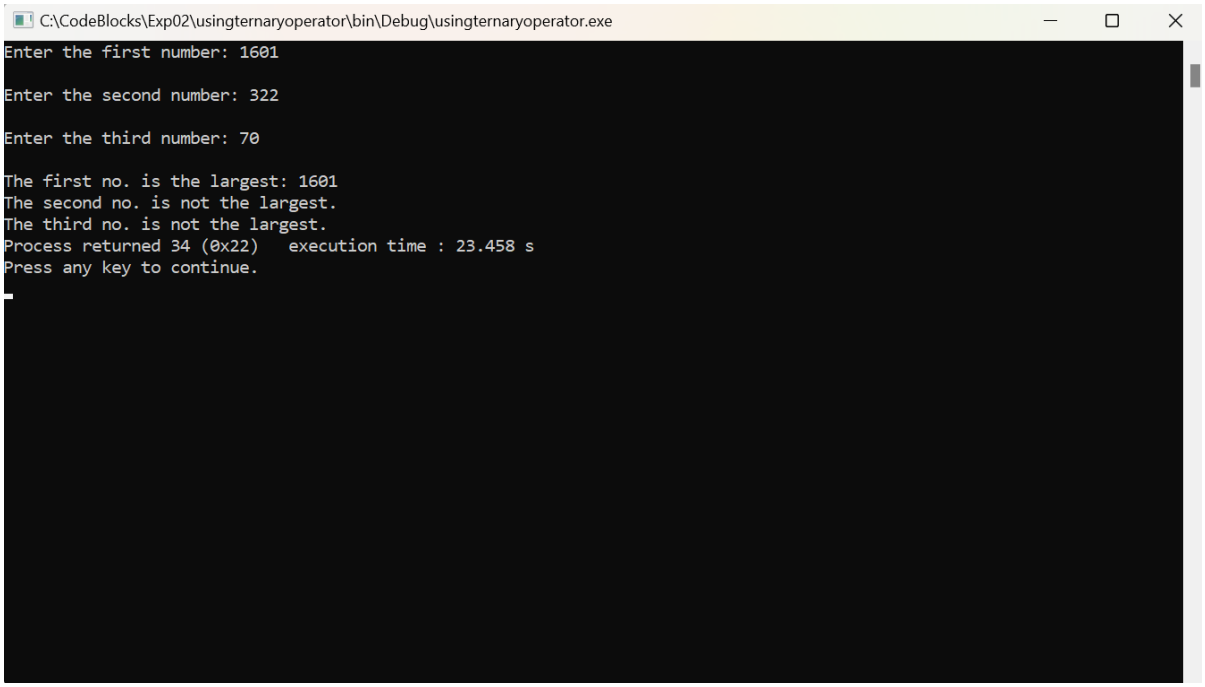
```
#include<stdio.h>

void main()
{
   int a,b,c;

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

   printf("\nEnter the second number: ");
   scanf("%d",&b);

   printf("\nEnter the third number: ");
   scanf("%d",&c);

   (a>b && a>c)? printf("\nThe first no. is the largest: %d",a):printf("\nThe first no. is not the largest.");

   (b>a && b>c)? printf("\nThe second no. is the largest: %d",b):printf("\nThe second no. is not the largest.");

   (c>b && c>a)? printf("\nThe third no. is the largest: %d",c):printf("\nThe third no. is not the largest.");
}
```

Output(s):

1.If-else if-else:



2.Using ternary operator:



Conclusion: The largest number was therefore correctly printed as the output, from the three numbers inputted by the user. The numbers that were inputted were compared using the nested if-else statement and the ternary operator in the code and then the largest number of them was thus found.

Post Lab Descriptive Questions:

1. Explain bitwise operators with examples.

Ans: Bitwise operator works on bits and performs bit-by-bit operation. The truth tables for &, |, and ^ is as follows −

		AND	OR	Exclusive OR EXOR
p	q	p & q	p | q	p ^ q
0	0	0	0	0
0	1	0	1	1
1	1	1	1	0
1	0	0	1	1

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) = 12, i.e., 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) = 61, i.e., 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) = 49, i.e., 0011 0001
~	Binary One's Complement Operator is unary and has the effect of 'flipping' bits.	(~A) = ~(60), i.e,. -0111101
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 = 240 i.e., 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 = 15 i.e., 0000 1111

2. Write a code snippet to perform left shifting of bits by some positions.

Ans: #include <stdio.h>

```
void main()
{
 int x = 60; /* 60 = 0011 1100 */

 int y = 0;

 y = x << 2; /* 240 = 1111 0000 */

 printf("Value of y is: %d\n", y );
 }
```

3. Write associative rules and the precedence table of various operators.

Ans:

Operator	Description of Operator	Associativity
.	Direct member selection	Left to right
->	Indirect member selection	Left to right
[]	Array element reference	Left to right
()	Functional call	Left to right
~	Bitwise(1's) complement	Right to left
!	Logical negation	Right to left
–	Unary minus	Right to left
+	Unary plus	Right to left
—	Decrement	Right to left
++	Increment	Right to left
*	Pointer reference	Right to left
&	Dereference (Address)	Right to left
(type)	Typecast (conversion)	Right to left
sizeof	Returns the size of an object	Right to left

%	Remainder	Left to right
/	Divide	Left to right
*	Multiply	Left to right
–	Binary minus (subtraction)	Left to right
+	Binary plus (Addition)	Left to right
>>	Right shift	Left to right
<<	Left shift	Left to right
>	Greater than	Left to right
<	Less than	Left to right
>=	Greater than or equal	Left to right
<=	Less than or equal	Left to right
==	Equal to	Left to right
!=	Not equal to	Left to right
^	Bitwise exclusive OR	Left to right
&	Bitwise AND	Left to right

||	Logical OR	Left to right
|	Bitwise OR	Left to right
?:	Conditional Operator	Right to left
&&	Logical AND	Left to right
,	Separator of expressions	Left to right
=	Simple assignment	Right to left
/=	Assign quotient	Right to left
*=	Assign product	Right to left
%=	Assign remainder	Right to left
-=	Assign difference	Right to left
+=	Assign sum	Right to left
|=	Assign bitwise OR	Right to left
^=	Assign bitwise XOR	Right to left
&=	Assign bitwise AND	Right to left
>>=	Assign right shift	Right to left

4. What are different storage class specifiers in C?

Ans: A variable given in a C program will have two of the properties: storage class and type. Here, type refers to any given variable's data type, while the storage class determines that very variable's lifetime, visibility, and also its scope.

Class	Name of Class	Place of Storage	Scope	Default Value	Lifetime
auto	Automatic	RAM	Local	Garbage Value	Within a function
extern	External	RAM	Global	Zero	Till the main program ends. One can declare it anywhere in a program.
static	Static	RAM	Local	Zero	Till the main program ends. It retains the available value between various function calls.
register	Register	Register	Local	Garbage Value	Within the function

Date: _____ **Signature of faculty in-charge**