TASK NO 3

Q 1: Can you explain what relational operators are in programming, and what type of output they produce?

ANS: Relational operators are used to compare numerical data. They check for a particular relationship between these values if a relationship exists, then it returns a Boolean true (1) otherwise it returns a Boolean false (0).

Here are the most commonly used relational operators

OPERATOR	DESCRIPTION	EXAMPLE
< (Less than)	This operator checks whether the first value is **less** than second value or not then it yields a Boolean true(1) otherwise false(0).	3<4; This will be true because 3 is less than 4.
>(greater than)	This operator checks whether the first value is **greater** than second value or not then it yields a Boolean true(1) otherwise false(0)	3>4; This will be false because 3 is not greater than 4.
<= (less than or equal to)	This operator checks whether the first value is **less than** or **equal to** second value or not then it yields a Boolean true(1) otherwise false(0)	3<=4; This will be true because 3 is less than 4.
>= (greater than or equal to)	This operator checks whether the first value is **greater than** or **equal to** second value or not then it yields a Boolean true(1) otherwise false(0)	3>=4; This will be false because 3 is not greater or equal to 4.

Q 2: How about logical operators – what do they do and what type of output do they produce?

ANS: Logical operators are used to combine or modify Boolean expressions. When more than one relational operator is used simultaneously then logical operators are used in getting the simultaneous relation.

The most common logical operators are as follows:

1. **OR (||):** This operator returns **true** (1) if either one of the two expressions is true(1) .
   (2<3)||(6<=4)
   The first expression results in true while second expression results in false so the overall expression evaluates to **true**.

2. **AND (&&):** This operator returns **true** (1) if both of the expressions are true (1).
   (2<3)&&(6<=4)
   The first expression results in true while second expression results in false so the overall expression evaluates to **false**.

3. **NOT (!):** This operator returns **false** (0) if expression is true (1), and vice versa. It only uses a single Boolean expression.
   !(2<3)
   The above expression results in true so the operator returns **false.**

Q 3: In what situations do we typically use relational and logical operators in programming?

ANS: Relational and logical operators are typically used in programming for decision making and flow control. Relational and equality operators usually compare two numbers and return a value of true or false (forming a logic expression). Logical operators combine logical values of true or false into a logical expression.

1. **Conditional statements:** Relational and logical operators are used to evaluate conditions in if-else statements, switch statements, and ternary operators to determine the flow of execution.
2. **Looping constructs:** Relational operators are used in while, do-while, and for loops to control the iteration of the loop based on a specified condition.
3. **Boolean expressions:** Logical operators are used to combine or modify Boolean expressions to create complex conditions that can be evaluated in conditional statements and looping constructs.
4. **Sorting and searching:** Relational operators are used in sorting and searching algorithms to compare elements and determine their order or position.
5. **Input validation:** Relational operators are used to validate user input in order to ensure that it meets certain criteria or constraints.

Overall, relational and logical operators are fundamental building blocks of many programming languages and are essential tools for developers when it comes to controlling the flow of execution in their programs.

Q 4: How do we use decision making in C++ programming?

ANS: Decision making structures require that the programmer specify one or more conditions to be evaluated or tested by the program, along with a statement or statements to be executed if the condition is determined to be true, and optionally, other statements to be executed if the condition is determined to be false.

1. **IF-ELSE Statement:** It is used to choose between certain blocks based on the relational expression if it is true or false. Both have a separate block and only one of them will be executed. The syntax is given as:

```
if(relational expression)
{
//body of if statement
statements to be executed if relational expression is true
}
else
{
//body of else statement
statements to be executed if relational expression is false
}
```

2. **SWITCH Statements:** The switch statement evaluates a variable and compares it to a series of values using the "case" keyword. If the variable matches one of the values, the corresponding block of code is executed. If the variable doesn't match any of the values, the "default" block of code is executed. The syntax is given as:

```
switch (variable) {
case value1:
// code to execute if variable equals value1
break;
case value2:
// code to execute if variable equals value 2
break;
default:
// code to execute if variable doesn’t match any of the values
break;}
```

Q 5: What is an If statement in C++, and how is it used in decision making?

ANS: The If statement evaluates the condition inside the parentheses ( ). If the condition evaluates to true, the code inside the body of it is executed. If the condition evaluates to false, the code inside the body of it is skipped.

The basic syntax of an If statement is given as:

```
If (condition)  {
// code to execute if condition is true
}
```

In this program, the user is prompted to enter a number, and the program then checks whether the number is greater than zero using an if statement. If the number is greater than 0, then the output will be “The number is positive”.

```
#include <iostream>
using namespace std;
int main()
{
  int num;
 cout << "Enter a number: ";
  cin >> num;
  if (num > 0) {
    cout << "The number is positive." << endl;}
  return 0;
}
```