Decision-making

Decision-making is about deciding the order of execution of statements based on certain conditions. ***Visual Studio Dot Net C#*** language handles decision-making by supporting the following statements.

When you write a computer program, regardless of the programming language, you often need to execute different set of statements depending on some satisfied condition. The process of determining the order in which statements execute in a program is called decision-making or flow of Control. The most common type of decision statement is the ***if*** statement which you study following Chapter.

1. ***if*** statement
2. ***if, else*** *statement*
3. ***if, else if. else*** *statement*
4. ***switch, case, default*** statement
5. **conditional operator** statement

1. Decision making with if statement

1. if statements

The ***if*** statement may be implemented in different forms depending on the evaluation of a conditional expression return a value of **True**. The syntax for a simple ***if*** statement is as follows:

Syntax: ***For one statement without block or curly braces.***

Flowchart of if statement

```
if (conditional expression)
    Statement;
```

Syntax: For block or more than one statement.

```
if (conditional expression)
{   Statement1;     Statement2;
    :
}
```

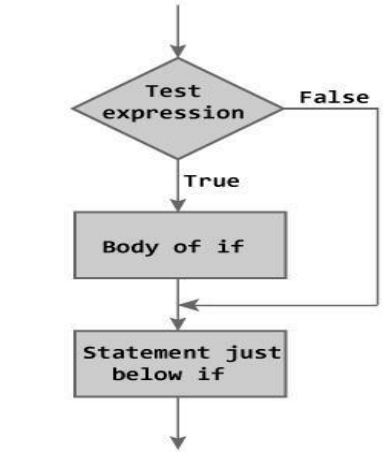
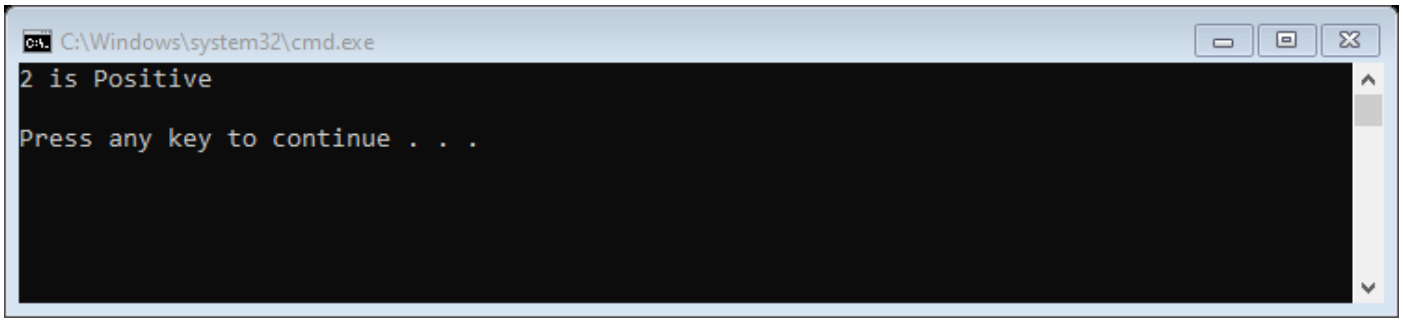


Figure: Flowchart of if Statement

If the expression is evaluated and found to be true, the single statement following the "**if**" is executed. **if** false, the following statement is skipped. Here a compound statement composed of several statements bounded by braces can replace the single statement.

Example *if* condition is True than execute For one statements.

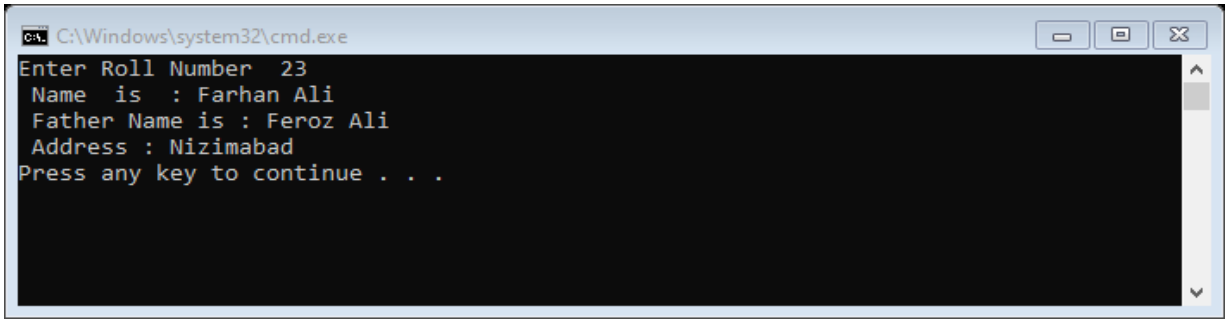
```
using System;

namespace Condition2
{
    class Program
    {
        static void Main(string[] args)
        {

            int Number = 2;
            if (Number > 0)
                Console.WriteLine(Number +" is Positive \n");


        }

    }
}
```



Example *if* condition is true than execute for block or more than one statements.

```
using System;

namespace Condition2
{
    class Program
    {
        static void Main(string[] args)
        {


            short Roll;
            Console.Write("Enter Roll Number  ");
            Roll = Convert.ToInt16(Console.ReadLine());

            if (Roll == 21)
            {
                Console.WriteLine(" Name  is  : Nadeem Mustafa  ");
                Console.WriteLine(" Father Name is : Fahim Mustafa ");
                Console.WriteLine(" Address : F.B.Area Karachi  ");
            }
```

```
        if (Roll == 23)
        {
            Console.WriteLine(" Name  is  : Farhan Ali  ");
            Console.WriteLine(" Father Name is : Feroz Ali ");
            Console.WriteLine(" Address : Nizimabad ");
        }

    }
 }
}
```



2. if, else statement:

This feature permits the programmer to write a single comparison, and then execute one of the two statements depending upon whether the test expression is true or false. The general form of the **if-else** statement is.

Syntax:

```
if( expression )
    statement1 ;
else
   statement2 ;
```

Flowchart of if...else statement

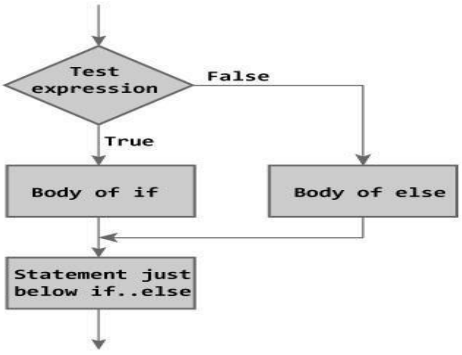
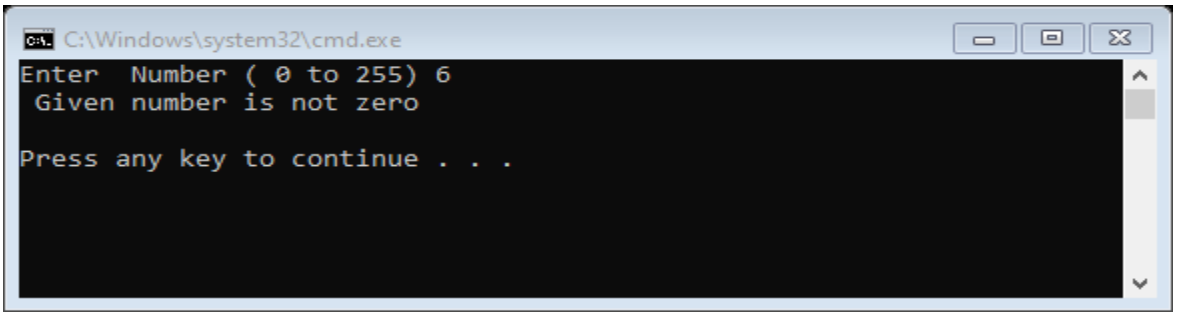


Figure: Flowchart of if...else Statement

Here also expression in parentheses must evaluate to (a Boolean) true or false. Typically, you are testing something to see **if** it's true, and then running a code block (one or more statements) **if** it is true, and another block of code **if** it isn't. The statement1 or statement2 can be either simple or compound statement.

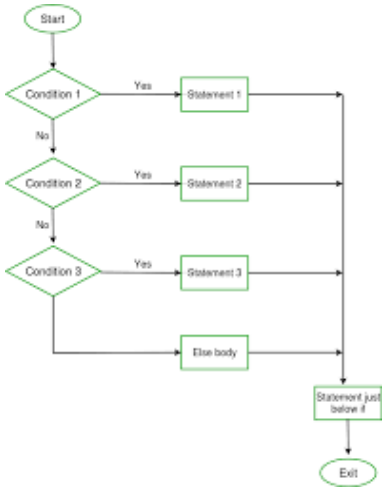
The following program demonstrates a legal **if else** statement:

```
using System;
namespace Condition2
{
    class Program
    {
        static void Main(string[] args)
        {
            short Number;
            Console.Write("Enter  Number ( 0 to 255 ) ");
            Roll = Convert.ToByte(Console.ReadLine());

             if (   Number == 0 )
                   Console.WriteLine(" Given number is Zero \n");
             else
                   Console.WriteLine(" Given number is not zero  \n");
        }
    }}
```



3. if .. else if.. else:

This brings up the other **if-else** construct; **if**, **else if**, **else**. This construct is useful where more than two alternatives or options are available for selection.



Syntax:

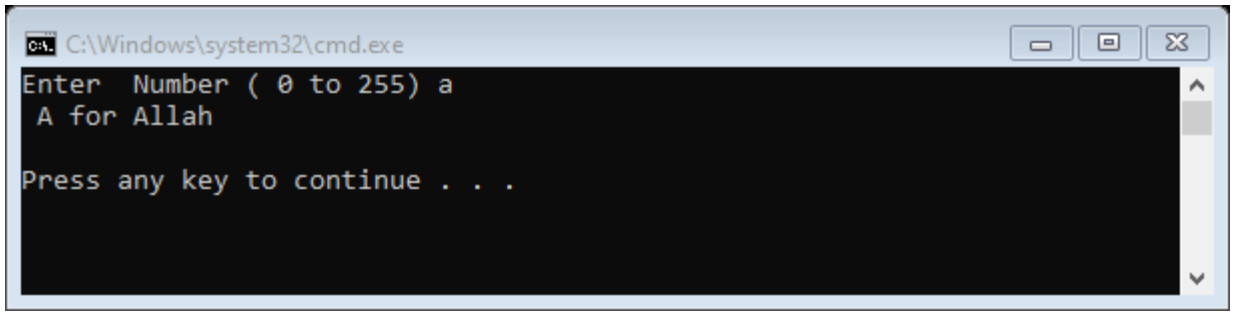
if (condition-1)
    statement-1;
else if (condition-2)
    statement-2;
else if(condition-3)
    statement-3;
else
    statements-4 ;

The various conditions are evaluated one by one starting from top to bottom, on reaching a condition evaluating to true the statement group associated with it are executed and skip other statements. If none of expression is evaluate to true, then the statement or group of statement associated with the final **else** is executed.

The following program demonstrates a legal **if- else if -else** statement:

```
using System;

namespace Condition2
{
    class Program
    {
        static void Main(string[] args)
        {
            char alphabet;
            Console.Write("Enter  Number ( 0 to 255) ");
            alphabet = Convert.ToChar( Console.ReadLine());

            if (alphabet == 'A' || alphabet == 'a')
                Console.WriteLine(" A for Allah  \n ");
            else if (alphabet == 'B' || alphabet == 'b')
                Console.WriteLine("  B for Bissmillah \n");
            else if (alphabet == 'C' || alphabet == 'c')
                Console.WriteLine(" \n C for Captain \n");

            else
                Console.WriteLine(" Unknown Character \n");
        }

    }
    }
```

Output will be depending on character value in alphabet variable: -



Difference between *if* -- *else if* – *else* and simple *if* is that:

if Statement

```
using System;

namespace Condition3
{
    class Program
    {
        static void Main(string[] args)
        {
              byte per;

       Console.Write("Enter  Percentage : ");
       per = Convert.ToByte(Console.ReadLine());


           if(  per >=80)
              Console.WriteLine(" Grade A+ ");
           if (  per>=70 )
              Console.WriteLine(" Grade A  ");
           if (  per>=60)
              Console.WriteLine(" Grade B  ");
          if (  per>=50)
              Console.WriteLine("Grade C ");
       }
    }
}
```

Output:

Grade A Grade B Grade C

// it will checks all conditions and execute all conditions that are true//

if else if Statement

```
using System;
namespace Condition3
{
    class Program
    {
        static void Main(string[] args)
        {
              byte per;

           Console.Write("Enter  Percentage : ");
           per = Convert.ToByte(Console.ReadLine());


           if(  per >=80)
              Console.WriteLine(" Grade A+ ");
           else if (  per>=70 )
              Console.WriteLine(" Grade A  ");
           else if (  per>=60)
              Console.WriteLine(" Grade B  ");
          else if (  per>=50)
              Console.WriteLine("Grade C ");

       }
    }
}
```

Output:

Grade B

// It stop when the condition is true (i.e. not check further) and then exit from the **if** – structure//

Nested If Expression:

If there is another structure within *if* structure that is called nested **if** statement.

Syntax:

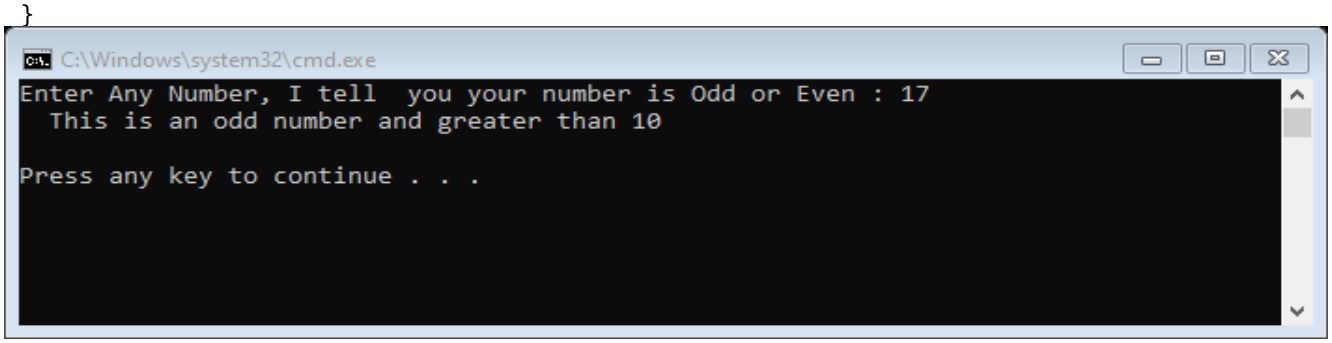
```
if( expression1 )  //                          ←——        Outer if
 {
      if( expression2 )    //          ← Inner If
     {
        statement Block1;
     } // end of inner if
   else    // ← else of inner if
    {
        statement   Block2
    }
}←——— // Outer if ends here
else  ←//  else of outer if
{
    statement block3;
}
```

if 'expression1' is false the **'statement-block3'** will be *executed*, otherwise it continues to perform the test for 'expression1' . ***If** the 'expression2' is **true*** the **'statement-block1'** is *executed* otherwise 'statement-block2' is executed.

Example

The **if-else** statement can also use to test for Nested conditions. The following example uses two conditions so that **if** the first test fails, we want to perform a second test before deciding what to do:

```
namespace Condition3
{
    class Program
    {
        static void Main(string[] args)
        {
              short n;
              Console.Write("Enter Any Number, I tell  you your number is Odd or Even : ");
              n =  Convert.ToInt16(Console.ReadLine());
            if (  n %2 != 0)
            {
                 if (  n>10)
                 {
                      Console.WriteLine("  This is an odd number and greater than 10 \n");
                 }else
                       Console.WriteLine("  this is an odd number and less than 10 \n" );
             }
            else
           {
               Console.WriteLine(" this is an even number \n");
           }

       }
    }
   }
```



4. Switch statement

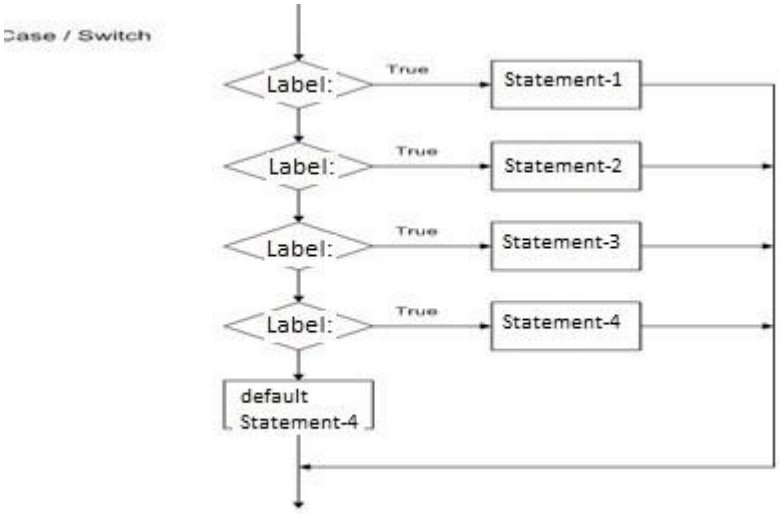
Switch statement is used to solve multiple option type. Another condition Visual Studio Dot Net C# statement that is used for controlling program flow is the **switch** statement. A switch statement, **in C#, is a selection statement that allows for the transfer of program control to a statement list with a switch label that corresponds** to the value of the switch expression. A ***switch*** statement consist of the following components: the keyword ***switch***, an expression, an opening brace, one or more ***case*** statements, a ***default*** label, and a closing brace. A **case** statement

consists of a ***case*** label, the executable statements, and the keyword ***break*** use for exit the **switch** block. The syntax for the **switch** statement is as follows:

Syntax:

```
switch(expression)
{
    case  label-1:
        statement-1;
        break;
    case label-2:
          satement-2;
          break;
    case label-3:
          statement-3;
          break;
    default:
          statement-otherwise-default;
}
```

Flow Chart of Switch Case default

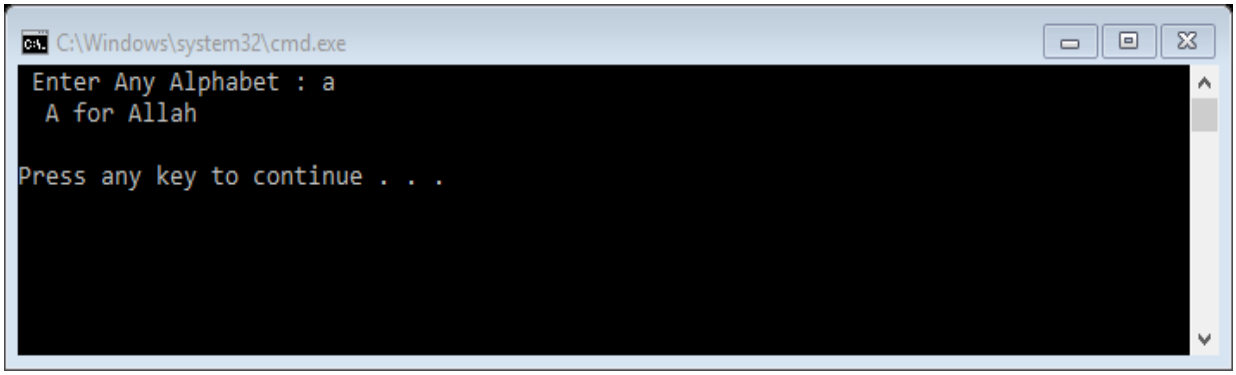


A **case** label consists of the keyword **case**, followed by a literal value or variable name It could be only character and integer followed by a colon. Visual Studio Dot Net C# compares the value returned from the **switch** statement expression to the literal value or value of the variable named following the **case** keyword. If a match is found, the statement following the **case** label statement executes.

Example:

```
using System;

namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
         char key;
         Console.Write(" Enter Any Alphabet : ");
         key= Convert.ToChar(Console.ReadLine());
         switch( key)
         {
            case 'A':
            case 'a':
                Console.WriteLine("  A for Allah  \n");
                break;
             case 'B':
             case 'b':
                Console.WriteLine(" B for Bissmillah    \n" );
                break;
```

```
            case 'C':
            case 'c':
               Console.WriteLine("C for Captain  \n ");
                break;
            default:
                  Console.WriteLine("  Unknown Value  \n ");
                  break;
        }

    }
  }
}
```



Points to Remember

- ✓ It isn't necessary to use **break** after each block, but **if** you do not use it, all the consecutive block of codes will get executed after the matching block.
- ✓ Switch **case** is valid only for "equals to" condition, it doesn't work for any other relational operators.
- ✓ A switch expression or case label must be ***bool, char ,string, integral, enum*** or corresponding null able type
- ✓ ***break*** keyword must be write with every case;

5. The Conditional (? :) Operator

We have covered **conditional operator ? :** in the previous condition expression which can be used to replace **if...else** statements. It has the following general form:

Syntax:

(Exp1) ?Exp2 : Exp3;

Or

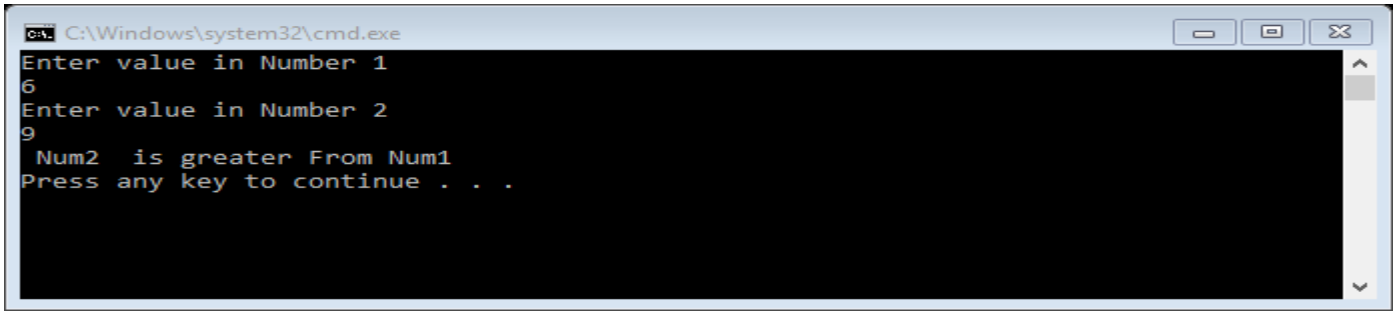
(Condition)? True : False;

Where Exp1, Exp2, and Exp3 are expressions. Notice the use and placement of the colon. The value of a **?** expression is determined like:

- Exp1 is evaluated. If it is true, then Exp2 is evaluated and becomes the value of the entire? Expression.
- If Exp1 is false, then Exp3 is evaluated and its value becomes the value of the expression.

Example:

```
using System;

namespace Condition4
{
    class Program
    {
        static void Main(string[] args)
        {
          int Num1 ,Num2;
          string result;

          Console.WriteLine("Enter value in Number 1");
          Num1 = Convert.ToInt16(Console.ReadLine());
          Console.WriteLine("Enter value in Number 2");
          Num2 = Convert.ToInt16(Console.ReadLine());

          result = (Num1 > Num2) ? " Num1 is greater From Num2 " : " Num2  is greater From Num1";
          Console.WriteLine(result);

        }
    }
}
```



It will check ***if*** *(Num1> Num2)* then it will evaluate *Expression 1* i.e. , but since Num1 is not greater than from Num2 here so it will evaluate *expression 2* i.e. **Points to Remember**

✓ It is also called **"Ternary Operator"** as it works on three operands.

Exercise

Theory Questions

1. Decision-making structures cannot be nested. True or false with give any example.
2. How do you perform more than one statement when a condition is true?
3. Correct the syntax error line by line.

```
static void Main(string[] args)
  {
1)    If   x > 5 ;
2)    {
3)       Y = x;
4)    else
5)       %y=  z
6)    }
  }
```

4. What function of ***break*** keyword/statement and where we can use it?
5. What is nested condition and Write syntax of nested condition.

Practical Questions

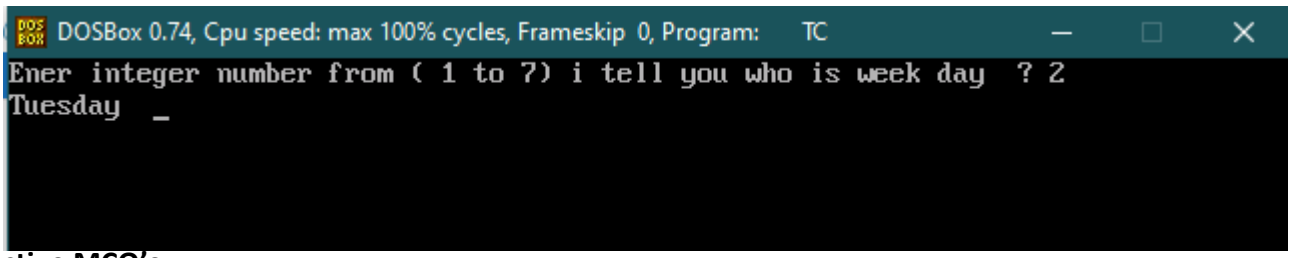
1. Write a simple program to construct a **Calculator** that can perform **Basic operation** Mathematical operation as well.
2. Write a simple program to check an alphabet entered by user is a ***Vowel*** or not **using if statement**, **if - else if- else statement** and **switch- case- default statements**.

if Statement	if else if Statement	switch case

3. Write a code to check an integer number entered by user is **Even** or **Odd** using **Conditional operator** statement.
4. Write a program to input subject marks Math, English and Physics then to prepare marks sheet of a student that will show the Obtain marks, percentage and grade.
5. Write a program that input a number, then report whether the number is in the range from 1 to 100. Otherwise the number above to 100.
6. The following is supposed to cause an action or print message, whenever x is 5 and y is 9.

```
static void main()
{  if ( x >=  y )
        Console.WriteLine("x is greater than y");
   else
        Console.WriteLine("y is greater than x ");
}
```

7. Write a **switch** statement that output messages indicating what day has been numerically input (for example 1-> Monday, 2-> Tuesday, 3-> Wednesday,...)



Objective MCQ's

1. You can exit a **switch** statement using a(n) ___
   a) break
   b) end
   c) quit
   d) complete

2. When the value return by a **switch** statement expression does not match a **case** label, the statements within the___ label execute.
   a) Exception
   b) Else
   c) Error
   d) default

3. In a simple ***if*** statement with no ***else***, what happens ***if*** the condition following the ***if*** is false?
   a) The program search for the last else in the program.
   b) Nothing, or control falls through to the statements following the if.
   c) The body of the if statement is executed.
   d) The program as a whole is executed.

4. The conditional operator statement that

```
static void Main(string[] args)
 {     int x=0 ;
   string result;
   result=  ( x ==0 )? " x equal to zero"): " x not equal to zero"  ;
 }
```

   a) Is incorrect syntax
   b) Is correct syntax, but x equal to zero will be print
   c) Cause a run time error
   d) Has no effect on the program

5. What value is assign in income tax to salary by the *if* statement when salary is 55000?

```
static void Main(string[] args)
{
  long salary =55000;
    if ( salary >70000 )
          Income_tax= 1000;
   else if ( salary > 50000 )
         Income_tax = 500;
    else
          Income_tax = 0;
 }
```

a) 1000
b) 500
c) 0
d) Nothing above all

6. Which is type of ternary operator?
   a) Logical operator
   b) Assignments operator
   c) Relational
   d) Conditional operator