Lesson 1

Introduction to Programming

Knowledge Assessment

Multiple Choice

Circle the letter that corresponds to the best answer.

1. Review the following code snippet:

   ```
   int n = 20;
   int d = n++ + 5;
   ```

   What will be the value of d after this code snippet is executed?

   a. 25
   b. 26
   c. 27
   d. 28

2. Review the following code snippet:

   ```
   private static void WhileTest()
   {
       int i = 1;
       while (i < 5)
       {
           Console.WriteLine("The value of i = {0}", i);
           i++;
       }
   }
   ```

   How many times will the while loop be executed in this code snippet?

   a. 0
   b. 1
   c. 4
   d. 5

3. Review the following code snippet:

   ```
   int number1 = 10;
   int number2 = 20;
   if (number2 > number1)
   Console.WriteLine("number1");
   Console.WriteLine("number2");
   ```

   What output will be displayed after this code snippet is executed?

a.

number1

b.

number2

c.

number1

number2

d.

number2

number1

4. In a switch statement, if none of the case statements match the switch expression, then control is transferred to which statement?
   a. break
   b. continue
   c. default
   d. return
5. You need to write code that closes a connection to a database and you need to make sure this code is always executed regardless of whether an exception is thrown. Where should you write this code?
   a. Within a try block
   b. Within a catch block
   c. Within a finally block
   d. Within the Main method
6. You need to store values ranging from 0 to 255. You also need to make sure that your program minimizes memory use. Which data type should you use to store these values?
   a. byte
   b. char
   c. short
   d. int
7. If you don't have a base case in your recursive algorithm, you create an infinite recursion. An infinite recursion will cause your program to throw an exception. Which exception will your program throw in such a case?
   a. OutOfMemoryException
   b. StackOverflowException
   c. DivideByZeroException
   d. InvalidOperationException
8. You are learning how to develop repetitive algorithms in C#. You write the following method:

   private static void ForTest()

```
{
   for(int i = 1; i < 5;)
   {
      Console.WriteLine("The value of i = {0}", i);
   }
}
```

How many repetitions will the for loop in this code perform?

a. 0

b. 4

c. 5

d. Infinite repetitions

9. Which of the following C# features should you use to organize code and create globally unique types?

   a. Assembly

   b. Namespace

   c. Class

   d. Data type

10. You write the following code snippet:

    ```
    int[] numbers = {1, 2, 3, 4};
    int val = numbers[1];
    ```

    You also create a variable of the RectangleHandler type like this:

    ```
    RectangleHandler handler;
    ```

    What is the value of the variable val after this code snippet is executed?

    a. 1

    b. 2

    c. 3

    d. 4

Fill in the Blank

Complete the following sentences by writing the correct word or words in the blanks provided.

1. The **switch** statement selects for execution a statement list having an associated label that corresponds to the value of an expression.
2. The **do-while** loop tests the condition at the bottom of the loop instead of at the top.
3. The only operator that takes three arguments is the **?:** operator..
4. The **foreach** loop is the most compact way to iterate through the items in a collection.

5. On a 32-bit computer, a variable of int data type takes **two** bytes of memory.
6. To access the first element of an array, you use an index of **0**.
7. **Recursion** is a programming technique that causes a method to call itself in order to compute a result.
8. **Constants** are data fields or local variables whose value cannot be modified.
9. When an algorithm involves a large number of conditions, a(n) **decision table** is a compact and readable format for presenting the algorithm.
10. A(n) **flowchart** is a graphical representation of an algorithm.

Competency Assessment

Project 1-1: Converting a Decision Table into a C# Program

You are developing an invoicing application that calculates discount percentages based on the quantity of a product purchased. The logic for calculating discounts is listed in the following decision table. If you need to write a C# method that uses the same logic to calculate the discount, how would you write such a program?

Quantity < 10	Y	N	N	N
Quantity < 50	Y	Y	N	N
Quantity < 100	Y	Y	Y	N
Discount	5%	10%	15%	20%

1. Create a new Visual C# console application project named Discount.

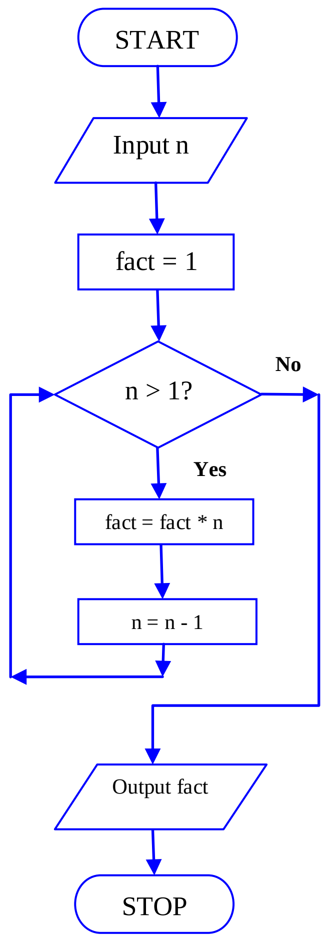
2. Replace the code in the class file with the following:

```
public class Discount
{
   static public void Main()
   {
      Console.Write("Please enter quantity: ");
      int quantity = Int32.Parse(Console.ReadLine());
      Console.WriteLine("Your discount is: {0}%",
         CalculateDiscount(quantity));
   }

   private static int CalculateDiscount(int quantity)
{
    int discount;

    if (quantity >= 100)
        discount = 20;
    else if (quantity >= 50)
        discount = 15;
    else if (quantity >= 10)
        discount = 10;
    else
        discount = 5;

    return discount;
}
}
```

3. Build and run the project. Enter a quantity and match the calculated discount with the decision table.

Project 1-2: Converting a Flowchart into a C# Program

You are developing a library of mathematical functions. You first develop the following flowchart describing the algorithm for calculating the factorial of a number. You need to write an equivalent C# program for this flowchart. How would you write such a program?



1. **Create a new Visual C# console application project named Factorial.**
2. **Replace the code in the class file with the following:**

```
public class Factorial
{
   static public void Main()
   {
      Console.Write("Please enter a number: ");
      int number = Int32.Parse(Console.ReadLine());
      Console.WriteLine("Factorial of {0} is: {1}",
         number, ComputeFactorial(number));
   }

   private static int ComputeFactorial(int number)
   {
      int factorial = 1;
      while (number > 1)
      {
         factorial *= number--;
      }

      return factorial;
   }
}
```

3. Build and run the project. Enter a number and match the results of the program with the dry run of the flowchart.

Proficiency Assessment

Project 1-3: Handling Exceptions

You are writing code for a simple arithmetic library. You decide to create a method named Divide that takes two arguments, x and y, and returns the value of x/y. You need to catch any arithmetic exceptions that might be thrown for errors in arithmetic, casting, or data type conversions. You also need to catch any other exceptions that may be thrown from the code. To address this requirement, you need to create properly structured exception-handling code. How would you write such a program?

1. Create a new Visual C# console application project named ExceptionTest.

2. Replace the code in the class file with the following:

```
class ExceptionTest
{
  public static void Main()
  {
         int x = 10;
         int y = 0;
         Console.WriteLine(Divide(x, y));
  }

  public static int Divide(int x, int y)
  {
    int results = 0;
    try
    {
      results = x/y;
    }
  catch (ArithmeticException e)
  {
    Console.WriteLine(
     "ArithmeticException Handler: {0}",
     e.ToString());
  }
  catch (Exception e)
  {
    Console.WriteLine(
     "Generic Exception Handler: {0}",
     e.ToString());
  }
         return results;
    }
}
```

3. Build and run the project.

Project 1-4: Creating a Recursive Algorithm

You are developing a library of utility functions for your application. You need to write a method that takes an integer and counts the number of significant digits in it. You need to create a recursive program to solve this problem. How would you write such a program?

1. **Create a new Visual C# console application project named Reverse.**
2. **Replace the code in the class file with the following:**

```
public class Reverse
{
   static public void Main()
   {
      Console.Write("Please enter a number: ");
      int number = Int32.Parse(Console.ReadLine());
      Console.WriteLine("The number {0} has {1} digits",
         number, CountDigits(number));
   }
```

```
    private static int CountDigits(int number)
    {
        if (number / 10 == 0)
            return 1;
        else
            return 1 + CountDigits (number / 10);
    }
}
```

3. Build and run the project. Enter a number, and check the number of digits calculated by the program.