Pseudocode	FORTRAN (original)	
INPUT number	10	FORMAT(I4)
	20	READ 1, INUM
INPUT divisor	30	READ 1, IDIVSR
intermediate <-- number	40	IMED = INUM
WHILE intermediate >= divisor	50	IF (INUM – IDIVSR) 80, 60, 60
intermediate <-- intermediate – divisor	60	IMED = IMED – IDIVSR
END WHILE	70	GOTO 5
remainder <-- intermediate	80	IREMR = IMED
OUTPUT remainder	90	PUNCH 1, IREMR

Pseudocode

```
INPUT number

INPUT divisor

intermediate <-- number

WHILE intermediate >= divisor

    intermediate <-- intermediate – divisor

END WHILE

remainder <-- intermediate

OUTPUT remainder
```

COBOL (probably buggy, modified from "Hello World" program found on the Internet)

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. MODULO.
000300
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100310     ACCEPT NUMBER.
100320     ACCEPT DIVISOR.
100330     MOVE NUMBER TO INTERMEDIATE.
100340     PERFORM UNTIL INTERMEDIATE < DIVISOR
100350         SUBTRACT DIVISOR FROM INTERMEDIATE
100360     END-PERFORM.
100400     MOVE INTERMEDIATE TO REMAINDER
100500     DISPLAY REMAINDER.
100600     STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800     EXIT.
```

Pseudocode	BASIC (interactive, from the original 1964 Dartmouth manual)	
INPUT number	10	DATA user_types_value_of_N
	15	READ N
INPUT divisor	20	DATA user_types_value_of_D
	25	READ D
intermediate <-- number	30	LET I = N
WHILE intermediate >= divisor	40	IF I >= D GO TO 70
intermediate <-- intermediate – divisor	50	LET I = I – D
END WHILE	60	GO TO 40
remainder <-- intermediate	70	LET R = I
OUTPUT remainder	80	PRINT R

Pseudocode

```
INPUT number

INPUT divisor

intermediate <-- number

WHILE intermediate >= divisor

    intermediate <-- intermediate – divisor

END WHILE

remainder <-- intermediate

OUTPUT remainder
```

C

```
#include <stdio.h>

void main(int argc, char* argv)
{
    int number, divisor, intermediate;

    scanf(“%d”, number);
    scanf(“%d”, divisor);

    intermediate = number;

    while (intermediate >= divisor)
    {
        intermediate = intermediate – divisor;
    }

    remainder = intermediate;

    printf(“%d”, remainder);
}
```

Pseudocode

```
INPUT number

INPUT divisor

intermediate <-- number

WHILE intermediate >= divisor

    intermediate <-- intermediate – divisor

END WHILE

remainder <-- intermediate

OUTPUT remainder
```

FORTRAN 77

```
PROGRAM MODULO
INTEGER NUMBER, DIVISOR
INTEGER INTERMEDIATE

READ *, NUMBER
READ *, DIVISOR

INTERMEDIATE = NUMBER

DO WHILE (INTERMEDIATE .GE. DIVISOR)

    INTERMEDIATE = INTERMEDIATE - DIVISOR

END DO

REMAINDER = INTERMEDIATE

PRINT *, REMAINDER
```

Pseudocode

```
INPUT number
INPUT divisor
intermediate <-- number
WHILE intermediate >= divisor
    intermediate <-- intermediate – divisor
END WHILE
remainder <-- intermediate
OUTPUT remainder
```

BASIC (structured, as used in early Personal Computers)

```
INPUT number
INPUT divisor
intermediate = number
WHILE intermediate >= divisor
    intermediate = intermediate - divisor
WEND
remainder = intermediate
PRINT remainder
```

Pseudocode

```
INPUT number

INPUT divisor

intermediate <-- number

WHILE intermediate >= divisor

    intermediate <-- intermediate – divisor

END WHILE

remainder <-- intermediate

OUTPUT remainder
```

C++

```
#include <iostream>

public class Modulo
{
    public static int mod(int number, int divisor)
    {
        int intermediate;

        intermediate = number;

        while (intermediate >= divisor)
        {
            intermediate = intermediate – divisor;
        }

        remainder = intermediate;

        return remainder;
    }
```

```
    public Modulo()
    {
    }
}

int main()
{
    int number, divisor;

    cin >> number;
    cin >> divisor;
    cout << Modulo.mod(number, divisor);
    return 0;
}
```

Pseudocode

```
INPUT number

INPUT divisor

intermediate <-- number

WHILE intermediate >= divisor

    intermediate <-- intermediate – divisor

END WHILE

remainder <-- intermediate

OUTPUT remainder
```

Java (not object oriented, console input and output)

```
import java.io.*;

public class Modulo
{
    public static void main(String[] args)
    {
        int number, divisor, intermediate, remainder;

        number = Integer.valueOf(System.console().readLine());
        divisor = Integer.valueOf(System.console().readLine());

        intermediate = number;

        while (intermediate >= divisor)
        {
            intermediate = intermediate – divisor;
        }

        remainder = intermediate;

        System.out.println(remainder);
    }
}
```

Pseudocode

```
INPUT number

INPUT divisor

intermediate <-- number

WHILE intermediate >= divisor

    intermediate <-- intermediate – divisor

END WHILE

remainder <-- intermediate

OUTPUT remainder
```

Java (fully object oriented, console input and output)

```
import java.io.*;

public class Modulo
{
    private int number, divisor;

    public int mod()
    {
        int intermediate;

        intermediate = number;

        while (intermediate >= divisor)
        {
            intermediate = intermediate – divisor;
        }

        remainder = intermediate;

        return remainder;
    }
```

```
    public void setNumber(int n)
    {
        if (n > 0 && n < 1000000)
            number = n;
    }
    public void setDivisor(int d)
    {
        if (d > 0 && n < 1000000)
            divisor = d;
    }

    public Modulo()
    {
        number = 0;
        divisor = 1;
    }

    public static void main(String[] args)
    {
        Modulo modulo = new Modulo();

        modulo.setNumber(
            Integer.valueOf(System.console().readLine()));
        modulo.setNumber(
            Integer.valueOf(System.console().readLine()));

        System.out.println(modulo.mod());
    }
}
```

Pseudocode

```
INPUT number

INPUT divisor

intermediate <-- number

WHILE intermediate >= divisor

   intermediate <-- intermediate – divisor

END WHILE

remainder <-- intermediate

OUTPUT remainder
```

php

```
<html>
<head>
<?php

   function modulo($n, $d)  {

      $i = $n;

      while  ($i  >=  $d)  {
         $i  =  $i - $d;
       }

       $r  =  $i;

      return $r;
   }
?>
</head>
```

```
<body>

<p>
Number:<input id='numberInput‘ type='integer'>
   </input><br/><br/>

Divisor:<input id='divisorInput' type='integer'>
    </input><br/><br/>

<button id='runModulo' onclick='modulo()'>
    Find remainder</button>
</p>

</body>
</html>
```

Pseudocode

```
INPUT number

INPUT divisor

intermediate <-- number

WHILE intermediate >= divisor

    intermediate <-- intermediate – divisor

END WHILE

remainder <-- intermediate

OUTPUT remainder
```

javascript (not related to Java)

```
<html>

<head>

<script type="text/javascript">

    function modulo()
    {
        var number = document.getElementById('numberInput').value;
        var divisor = document.getElementById('divisorInput').value;

        var intermediate = number;

        while (intermediate >= divisor)
        {
            intermediate = intermediate - divisor;
        }

        var remainder = intermediate;

        alert(remainder);
    }

</script>

 </head>
```

```
<body>

<p>
Number:<input id='numberInput‘ type='integer'>
   </input><br/><br/>

Divisor:<input id='divisorInput' type='integer'>
    </input><br/><br/>

<button id='runModulo' onclick='modulo()'>
    Find remainder</button>
</p>

</body>
</html>
```

Pseudocode

```
INPUT number

INPUT divisor

intermediate <-- number

WHILE intermediate >= divisor

    intermediate <-- intermediate – divisor

END WHILE

remainder <-- intermediate

OUTPUT remainder
```

Visual Basic (not object oriented, console input and output)

```
Module Modulo

    Sub Main()

        Dim number, divisor, intermediate, remainder As Int32

        Int32.TryParse(System.Console.In.ReadLine(), number);
        Int32.TryParse(System.Console.In.ReadLine(), divisor);

        intermediate = number;

        While (intermediate >= divisor)

            intermediate = intermediate – divisor;

        End While

        remainder = intermediate;

        System.Console.Out.WriteLine(remainder);

    End Sub

End Module
```

Pseudocode

```
INPUT number

INPUT divisor

intermediate <-- number

WHILE intermediate >= divisor

    intermediate <-- intermediate – divisor

END WHILE

remainder <-- intermediate

OUTPUT remainder
```

C# (not object oriented, console input and output)

```
using System;

class Program
{
    static void main(string[] args)
    {
        int number, divisor, intermediate, remainder;

        Int32.TryParse(System.Console.In.ReadLine(), out number);
        Int32.TryParse(System.Console.In.ReadLine(), out divisor);

        intermediate = number;

        while (intermediate >= divisor)
        {
            intermediate = intermediate – divisor;
        }

        remainder = intermediate;

        System.Console.Out.WriteLine(remainder);
    }
}
```

Visual Basic (fully object oriented windows)

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class Prime_Windows
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.Icontainer

 'NOTE: The following procedure is required by the Windows Form
Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
Private Sub InitializeComponent()
        Me.NumberInput = New System.Windows.Forms.TextBox
        Me.AnswerLable = New System.Windows.Forms.Label
        Me.SuspendLayout()
        '
        'NumberInput
        '
        Me.NumberInput.Location = New System.Drawing.Point(103, 47)
        Me.NumberInput.Name = "NumberInput"
        Me.NumberInput.Size = New System.Drawing.Size(87, 20)
        Me.NumberInput.TabIndex = 0
        '
        'AnswerLable
        '
        Me.AnswerLable.AutoSize = True
        Me.AnswerLable.Location = New System.Drawing.Point(108, 104)
        Me.AnswerLable.Name = "AnswerLable"
        Me.AnswerLable.Size = New System.Drawing.Size(0, 13)
        Me.AnswerLable.TabIndex = 1
        '
        'Prime_Windows
        '
```

```
Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(292, 273)
        Me.Controls.Add(Me.AnswerLable)
        Me.Controls.Add(Me.NumberInput)
        Me.Name = "Prime_Windows"
        Me.Text = "Prime"
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents NumberInput As System.Windows.Forms.TextBox
    Friend WithEvents AnswerLable As System.Windows.Forms.Label

End Class
```

```
Public Class Prime_Windows

    Private Sub NumberInput_KeyUp(ByVal sender As_
        System.Object, ByVal e As_
        System.Windows.Forms.KeyEventArgs) Handles_
        NumberInput.KeyUp

        Dim primeFinder As New Prime

        If e.KeyCode = Keys.Enter Then

            If Int32.TryParse(NumberInput.Text, number) = 0 Then
                answer = “Invalid input”
            ElseIf number < 0 Then
                answer = “Must be positive”
            Else
                If number >= 2 And
                    primeFinder.determineIfPrime(number) Then
                    answer = number.ToString + " is a prime."
                Else
                    answer = number.ToString + " is not a prime.“
                End If
            End If
            AnswerLable.Text = answer
        End If

    End Sub

    Dim number As Integer
    Dim answer As String

End Class
```

```
Public Class Prime

    Public Function determineIfPrime(ByVal number As Integer) As Boolean
        Dim remainder As Integer
        Dim divisor As Integer
        Dim prime As Boolean

        Dim finder As New ModuloFinder


        prime = True

        divisor = 2

        While divisor < number
            remainder = finder.modulo(number, divisor)

            If remainder = 0 Then
                prime = False
            End If

            divisor = divisor + 1

        End While

        Return prime

    End Function

End Class
```

Pseudocode

```
INPUT number

INPUT divisor

intermediate <-- number

WHILE intermediate >= divisor

    intermediate <-- intermediate – divisor

END WHILE

remainder <-- intermediate

OUTPUT remainder
```

```
Public Class ModuloFinder

    Public Function modulo(ByVal number As_
        Integer, ByVal divisor As Integer) As Integer

        Dim intermediate As Integer
        Dim remainder As Integer

        intermediate = number

        While intermediate >= divisor

            intermediate = intermediate – divisor

        End While

        remainder = intermediate

        Return remainder
    End Function

End Class
```

C# (fully object oriented windows)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace Prime_CSharp_Windows
{
   static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new PrimeWindow());
      }
   }
}
```

```
namespace Prime_CSharp_Windows
{
   partial class PrimeWindow
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources
      ///       should be disposed; otherwise, false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }
```

```
#region Windows Form Designer generated code

    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
        this.NumberInput = new System.Windows.Forms.TextBox();
        this.AnswerLabel = new System.Windows.Forms.Label();
        this.SuspendLayout();
        //
        // NumberInput
        //
        this.NumberInput.Location = new System.Drawing.Point(94, 57);
        this.NumberInput.Name = "NumberInput";
        this.NumberInput.Size = new System.Drawing.Size(100, 20);
        this.NumberInput.TabIndex = 0;
        this.NumberInput.KeyUp += new
            System.Windows.Forms.KeyEventHandler(this.NumberInput_KeyUp);
        //
        // AnswerLabel
        //
        this.AnswerLabel.AutoSize = true;
        this.AnswerLabel.Location = new System.Drawing.Point(94, 109);
        this.AnswerLabel.Name = "AnswerLabel";
        this.AnswerLabel.Size = new System.Drawing.Size(0, 13);
        this.AnswerLabel.TabIndex = 1;
//
        // PrimeWindow
        //
        this.AutoScaleDimensions = new
            System.Drawing.SizeF(6F, 13F);
        this.AutoScaleMode =
            System.Windows.Forms.AutoScaleMode.Font;
        this.ClientSize = new System.Drawing.Size(292, 273);
        this.Controls.Add(this.AnswerLabel);
        this.Controls.Add(this.NumberInput);
        this.Name = "PrimeWindow";
        this.Text = "Prime";
        //this.Load += new
            System.EventHandler(this.PrimeWindow_Load);
        this.ResumeLayout(false);
        this.PerformLayout();

    }

    #endregion

    private System.Windows.Forms.TextBox NumberInput;
    private System.Windows.Forms.Label AnswerLabel;
  }
}
```

```
using System;
using System.Windows.Forms;

namespace Prime_CSharp_Windows
{
   public partial class PrimeWindow : Form
   {
      public PrimeWindow()
      {
         InitializeComponent();
      }

      private void NumberInput_KeyUp(object sender, KeyEventArgs e)
      {
         if (e.KeyCode == Keys.Enter)
         {
            if (Int32.TryParse(NumberInput.Text, out number) == false)
                answer = “Invalid input”;
            else if (number < 0)
                answer = “Must be positive”;
            else
            {
               if (Prime.determineIfPrime(number)
                  answer = number + " is a prime.";
               else
                  answer = number + " is not a prime.";
            }
            AnswerLabel.Text =  answer.ToString();
         }
      }
      private int number;
      private string answer;
   }

   public static class Prime
   {
      public static Boolean determineIfPrime(int number)
      {
         int remainder;
         int divisor;
         Boolean prime;

         prime = true;

         divisor = 2;

         while (divisor < number)
         {
            remainder = ModuloFinder.modulo(number, divisor);

            if (remainder == 0)
               prime = false;

            divisor = divisor + 1;
         }

         return prime;
      }
   }
```

Pseudocode

```
INPUT number

INPUT divisor

intermediate <-- number

WHILE intermediate >= divisor

   intermediate <-- intermediate – divisor

END WHILE

remainder <-- intermediate

OUTPUT remainder
```

```
public static class ModuloFinder
{
   public static int modulo(int number, int divisor)
   {
      int intermediate;
      int remainder;

      intermediate = number;

      while (intermediate >= divisor)
      {
         intermediate = intermediate – divisor;

      }

      remainder = intermediate;

      return remainder;
   }
  }
}
```