Visual Basic/C# Programming (330)

REGIONAL – 2017

Production Portion:

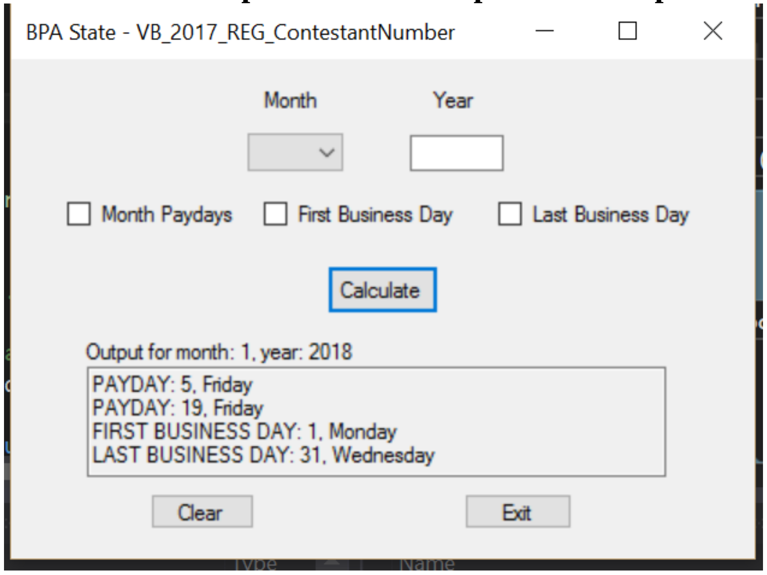
Program 1: Calendar Analysis	______________	(400 points)
TOTAL POINTS	______________	***(400 points)***

Judge/Graders: Please double check and verify all scores and answer keys!

Date & Time: Calendar

Test Cases

1. **Enter month of 1, year of 2018, click <Calculate> button**
   a. **EXPECTED RESULT: Error warning, Figure 3 from Test**
2. **Enter month of 1, year of 5000, check 'Month Paydays', click <Calculate> button**
   a. **EXPECTED RESULT: Error dialog, Figure 2 from Test**
3. **Enter month of 1, year of 2018, check all 3 checkboxes**
   a. **EXPECTED RESULT: Output label and output textbox updated as below**



```
Output for month: 1, year: 2018
PAYDAY: 5, Friday
PAYDAY: 19, Friday
FIRST BUSINESS DAY: 1, Monday
LAST BUSINESS DAY: 31, Wednesday
```

4. **Enter month of 1, year of 2018, check all 3 checkboxes. Click <clear> button**
   a. **EXPECTED RESULT: All fields are cleared and output label is returned to previous state.**
5. **Click <Exit> button**
   a. **EXPECTED RESULT: Figure 5 from test document**
6. **Additional valid tests:**
   a. **Month = 5, Year = 2019, Check 'Month Paydays'**
      i. **EXPECTED RESULT: 'PAYDAY', with Friday the 3rd and 17th**
   b. **Month = 2, Year = 2017, Check 'First Business Day'**
      i. **EXPECTED RESULT: 'FIRST BUSINESS DAY: 1, Wednesday'**
   c. **Month = 10, Year = 2020, Check 'Last Business Day'**
      i. **EXPECTED RESULT: : 'LAST BUSINESS DAY: 30, Friday**

Your application will be graded on the following criteria:

Solution and Project

Custom code is present	____ 10 points
All classes and methods are customized	____ 10 points

Program Design

Application GUI is designed according to specifications	____ 50 points

Program Execution

If program does not execute, then remaining items receive *partial credit* if credible code exists.

Program runs correctly	____ 50 points
Program produces correct output for payday calculation	____ 30 points
Program produces correct output for first business day	____ 30 points
Program produces correct output for last business day	____ 30 points
Program responds correctly for no selection	____ 10 points
Program responds correctly between each run	____ 10 points
Program exits correctly, with warning	____ 5 points
Program clear button works correctly	_____10 points

Source Code Review

Class code is commented, for each method, and as needed	____ 20 points
Code uses reasonable and consistent variable naming conventions	____ 20 points
Code correctly produces the menu with month and year	____ 20 points
Code exists to catch invalid date errors	____ 20 points
The code has a well-formed methods to process payday calculation	____ 25 points
The code has a well-formed methods to process first business day	____ 25 points
The code has a well-formed methods to process last business day	____ 25 points

Total Points: ________/ 400 points

C# Example Solution:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Collections.Specialized;
using System.Collections;

namespace BPA_MonthInfo_Regional
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        // Exit button click with close confirmation dialog
        private void button2_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Are you sure you want to exit this application?", "Close
Application", MessageBoxButtons.YesNo) == DialogResult.Yes)
            {
                this.Close();
            }
        }

        // Calculate answers based on user input
        private void button1_Click(object sender, EventArgs e)
        {

            // Must have valid values for month and year
            int numYear;
            int numMonth;
            bool isValidYear = int.TryParse(Year.Text, out numYear);
            bool isValidMonth = int.TryParse(monthSelect.Text, out numMonth);

            // Ensure at least one checkbox is selected, or alert the user
            if (!paydayCheckbox.Checked && !firstWorkdayCheckbox.Checked &&
!lastWorkdayCheckbox.Checked)
            {
                MessageBox.Show("You must select something to calculate", "Appication Warning",
                                    MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);
            }

            // clear output textfield
            outputText.Text = "";

            // If there is a valid month and year, continue processing
            if (isValidYear && isValidMonth && (numYear > 2010) && (numYear < 2300))
            {

                // Result string
                string resultString = "";

                // Was paydays selected
                if (paydayCheckbox.Checked)
                {
                    resultString += dateFunctions.calculatePayDays(numMonth, numYear);
                }


                // Was first workday of month selected
                if (firstWorkdayCheckbox.Checked)
                {
                    resultString += dateFunctions.firstWorkingDay(numMonth, numYear);
                }

                // Was last workday of month selected
                if (lastWorkdayCheckbox.Checked)
                {
                    resultString += dateFunctions.lastWorkingDay(numMonth, numYear);
                }

                // Now display results in output textfield
                outputText.Text = resultString;

            } else
            {
                outputText.Text = "ERROR: Must have valid month and year";
            }

            // set output label
            outputLabel.Text = "Output for month: " + monthSelect.Text + ", year: " + Year.Text;

            // clear fields
            paydayCheckbox.Checked = false;
            firstWorkdayCheckbox.Checked = false;
            lastWorkdayCheckbox.Checked = false;
            Year.Text = "";
            monthSelect.SelectedIndex = -1;

        }

        private void clearButton_Click(object sender, EventArgs e)
        {
            paydayCheckbox.Checked = false;
            firstWorkdayCheckbox.Checked = false;
            lastWorkdayCheckbox.Checked = false;
            Year.Text = "";
            monthSelect.SelectedIndex = -1;
            outputLabel.Text = "";
            outputText.Text = "";

        }
    }

    // This class has all of the calcuation functions called from the Calculate button
    class dateFunctions
    {
        // This method finds the paydays, the first and third Fridays of the month
        // Input: Month, Year
        // Output: Hashtable with two values (first and third day and date)
        static public string calculatePayDays(int month, int year)
        {

            // temporary output string to return
            string innerString = "";

            // Create data objects
            var firstOftargetMonth = new DateTime(year, month, 1);
            var firstOfNextMonth = firstOftargetMonth.AddMonths(1);

            // storing paydays in ordered dictionary,
            // but other data structures are fine
            OrderedDictionary payDays = new OrderedDictionary();
            int countFridays = 0;

            // iterate through month and identify first and third Fridays (paydays)
            for (DateTime date = firstOftargetMonth; date < firstOfNextMonth; date =
date.AddDays(1))
            {

                // Check if the incoming one is one of the paydays
                if (date.DayOfWeek.ToString().Equals("Friday"))
                {
                    // will this be our first or third (incoming)
                    if ((countFridays == 0) || (countFridays == 2))
                    {
                        payDays.Add(date.Day, date.DayOfWeek);
                    }

                    // increment our Friday counter
                    countFridays = countFridays + 1;
                }

            }

            // now build output string
            IDictionaryEnumerator myEnumerator = payDays.GetEnumerator();
            while (myEnumerator.MoveNext())
            {
                innerString += "PAYDAY: " + myEnumerator.Key + ", " +
                    myEnumerator.Value + System.Environment.NewLine;
            }

            // Return string with values
            return innerString;
        }

        // This method finds the first working day of the month
        // Input: Month, Year
        // Output: Hashtable with one value (first working day and date of month)
        static public string firstWorkingDay(int month, int year)
        {
            string innerString = "";

            var firstOftargetMonth = new DateTime(year, month, 1);
            var firstOfNextMonth = firstOftargetMonth.AddMonths(1);

            OrderedDictionary firstWorkingDay = new OrderedDictionary();
            bool notFound = true;

            for (DateTime date = firstOftargetMonth; date < firstOfNextMonth; date =
date.AddDays(1))
            {

                if ((notFound) && !(date.DayOfWeek.ToString().Equals("Saturday")) &&
                                !(date.DayOfWeek.ToString().Equals("Sunday")))
                {
                    notFound = false;
                    firstWorkingDay.Add(date.Day, date.DayOfWeek);
                }
            }

            // now build output string
            IDictionaryEnumerator myEnumerator = firstWorkingDay.GetEnumerator();
            while (myEnumerator.MoveNext())
            {
                innerString += "FIRST BUSINESS DAY: " + myEnumerator.Key + ", " +
                    myEnumerator.Value + System.Environment.NewLine;
            }


            return innerString;
        }


        // This method finds the last working day of the month
        // Input: Month, Year
        // Output: String with one key/value (last working day and date of month)
        static public string lastWorkingDay(int month, int year)
        {
            string innerString = "";

            var firstOftargetMonth = new DateTime(year, month, 1);
            var firstOfNextMonth = firstOftargetMonth.AddMonths(1);

            OrderedDictionary lastWorkingDay = new OrderedDictionary();

            for (DateTime date = firstOftargetMonth; date < firstOfNextMonth; date =
date.AddDays(1))
            {

                if (!(date.DayOfWeek.ToString().Equals("Saturday")) &&
                                !(date.DayOfWeek.ToString().Equals("Sunday")))
                {
                    if (lastWorkingDay.Count == 0)
                    {
                        lastWorkingDay.Add(date.Day, date.DayOfWeek);
                    }
                    if (lastWorkingDay.Count > 0)
                    {
                        lastWorkingDay.Clear();
                        lastWorkingDay.Add(date.Day, date.DayOfWeek);
                    }
                }
            }

            // now build output string
            IDictionaryEnumerator myEnumerator = lastWorkingDay.GetEnumerator();
            while (myEnumerator.MoveNext())
            {
                innerString += "LAST BUSINESS DAY: " + myEnumerator.Key + ", " +
                    myEnumerator.Value + System.Environment.NewLine;
            }

            // Return string with values
            return innerString;
        }
    }

}
```

Visual Basic Example Solution:

```
Imports System.Collections.Generic
Imports System.ComponentModel
Imports System.Data
Imports System.Drawing
Imports System.Linq
Imports System.Text
Imports System.Threading.Tasks
Imports System.Windows.Forms
Imports System.Collections.Specialized
Imports System.Collections

Public Partial Class Form1
       Inherits Form
       Public Sub New()
              InitializeComponent()
       End Sub

       ' Exit button click with close confirmation dialog
       Private Sub button2_Click(sender As Object, e As EventArgs)
              If MessageBox.Show("Are you sure you want to exit this application?", "Close
Application", MessageBoxButtons.YesNo) = DialogResult.Yes Then
                     Me.Close()
              End If
       End Sub

       ' Calculate answers based on user input
       Private Sub button1_Click(sender As Object, e As EventArgs)

              ' Must have valid values for month and year
              Dim numYear As Integer
              Dim numMonth As Integer
              Dim isValidYear As Boolean = Integer.TryParse(Year.Text, numYear)
              Dim isValidMonth As Boolean = Integer.TryParse(monthSelect.Text, numMonth)

              ' Ensure at least one checkbox is selected, or alert the user
              If Not paydayCheckbox.Checked AndAlso Not firstWorkdayCheckbox.Checked AndAlso Not
lastWorkdayCheckbox.Checked Then
                     MessageBox.Show("You must select something to calculate", "Appication
Warning", MessageBoxButtons.OKCancel, MessageBoxIcon.Warning)
              End If

              ' clear output textfield
              outputText.Text = ""

              ' If there is a valid month and year, continue processing
              If isValidYear AndAlso isValidMonth AndAlso (numYear > 2010) AndAlso (numYear <
2300) Then

                     ' Result string
                     Dim resultString As String = ""

                     ' Was paydays selected
                     If paydayCheckbox.Checked Then
                            resultString += dateFunctions.calculatePayDays(numMonth, numYear)
                     End If
```

```
                ' Was first workday of month selected
                If firstWorkdayCheckbox.Checked Then
                    resultString += dateFunctions.firstWorkingDay(numMonth, numYear)
                End If

                ' Was last workday of month selected
                If lastWorkdayCheckbox.Checked Then
                    resultString += dateFunctions.lastWorkingDay(numMonth, numYear)
                End If

                ' Now display results in output textfield

                outputText.Text = resultString
            Else
                outputText.Text = "ERROR: Must have valid month and year"
            End If

            ' set output label
            outputLabel.Text = "Output for month: " & monthSelect.Text & ", year: " &
Year.Text

            ' clear fields
            paydayCheckbox.Checked = False
            firstWorkdayCheckbox.Checked = False
            lastWorkdayCheckbox.Checked = False
            Year.Text = ""
            monthSelect.SelectedIndex = -1

        End Sub

        Private Sub clearButton_Click(sender As Object, e As EventArgs)
            paydayCheckbox.Checked = False
            firstWorkdayCheckbox.Checked = False
            lastWorkdayCheckbox.Checked = False
            Year.Text = ""
            monthSelect.SelectedIndex = -1
            outputLabel.Text = ""
            outputText.Text = ""

        End Sub
End Class

' This class has all of the calcuation functions called from the Calculate button
Class dateFunctions
        ' This method finds the paydays, the first and third Fridays of the month
        ' Input: Month, Year
        ' Output: Hashtable with two values (first and third day and date)
        Public Shared Function calculatePayDays(month As Integer, year As Integer) As String

            ' temporary output string to return
            Dim innerString As String = ""

            ' Create data objects
         Dim firstOftargetMonth As New DateTime(year, month, 1)
         Dim firstOfNextMonth As DateTime = firstOftargetMonth.AddMonths(1)

            ' storing paydays in ordered dictionary,
            ' but other data structures are fine
            Dim payDays As New OrderedDictionary()
            Dim countFridays As Integer = 0

            ' iterate through month and identify first and third Fridays (paydays)
            Dim [date] As DateTime = firstOftargetMonth
            While [date] < firstOfNextMonth

                ' Check if the incoming one is one of the paydays
                If [date].DayOfWeek.ToString().Equals("Friday") Then
                    ' will this be our first or third (incoming)
                    If (countFridays = 0) OrElse (countFridays = 2) Then
                        payDays.Add([date].Day, [date].DayOfWeek)
                    End If

                    ' increment our Friday counter
                    countFridays = countFridays + 1

                End If
                [date] = [date].AddDays(1)
            End While

            ' now build output string
            Dim myEnumerator As IDictionaryEnumerator = payDays.GetEnumerator()
            While myEnumerator.MoveNext()
                innerString += "PAYDAY: " & Convert.ToString(myEnumerator.Key) & ", " &
Convert.ToString(myEnumerator.Value) & System.Environment.NewLine
            End While

            ' Return string with values
            Return innerString
        End Function

        ' This method finds the first working day of the month
        ' Input: Month, Year
        ' Output: Hashtable with one value (first working day and date of month)
        Public Shared Function firstWorkingDay(month As Integer, year As Integer) As String
            Dim innerString As String = ""

         Dim firstOftargetMonth As New DateTime(year, month, 1)
         Dim firstOfNextMonth As DateTime = firstOftargetMonth.AddMonths(1)

         Dim firstWorkingDay2 As New OrderedDictionary()
            Dim notFound As Boolean = True

            Dim [date] As DateTime = firstOftargetMonth
            While [date] < firstOfNextMonth

                If (notFound) AndAlso Not ([date].DayOfWeek.ToString().Equals("Saturday"))
AndAlso Not ([date].DayOfWeek.ToString().Equals("Sunday")) Then
                    notFound = False
              firstWorkingDay2.Add([date].Day, [date].DayOfWeek)
                End If
                [date] = [date].AddDays(1)
            End While

            ' now build output string
         Dim myEnumerator As IDictionaryEnumerator = firstWorkingDay2.GetEnumerator()
            While myEnumerator.MoveNext()
                innerString += "FIRST BUSINESS DAY: " & Convert.ToString(myEnumerator.Key)
& ", " & Convert.ToString(myEnumerator.Value) & System.Environment.NewLine
```

```
            End While


            Return innerString
        End Function


        ' This method finds the last working day of the month
        ' Input: Month, Year
        ' Output: String with one key/value (last working day and date of month)
        Public Shared Function lastWorkingDay(month As Integer, year As Integer) As String
            Dim innerString As String = ""

         Dim firstOftargetMonth As New DateTime(year, month, 1)
         Dim firstOfNextMonth As DateTime = firstOftargetMonth.AddMonths(1)

         Dim lastWorkingDay2 As New OrderedDictionary()

            Dim [date] As DateTime = firstOftargetMonth
            While [date] < firstOfNextMonth

                If Not ([date].DayOfWeek.ToString().Equals("Saturday")) AndAlso Not
([date].DayOfWeek.ToString().Equals("Sunday")) Then
                If lastWorkingDay2.Count = 0 Then
                    lastWorkingDay2.Add([date].Day, [date].DayOfWeek)
                End If
                If lastWorkingDay2.Count > 0 Then
                    lastWorkingDay2.Clear()
                    lastWorkingDay2.Add([date].Day, [date].DayOfWeek)
                End If
                End If
                [date] = [date].AddDays(1)
            End While

            ' now build output string
         Dim myEnumerator As IDictionaryEnumerator = lastWorkingDay2.GetEnumerator()
            While myEnumerator.MoveNext()
                innerString += "LAST BUSINESS DAY: " & Convert.ToString(myEnumerator.Key) &
", " & Convert.ToString(myEnumerator.Value) & System.Environment.NewLine
            End While

            ' Return string with values
            Return innerString
        End Function
End Class
```