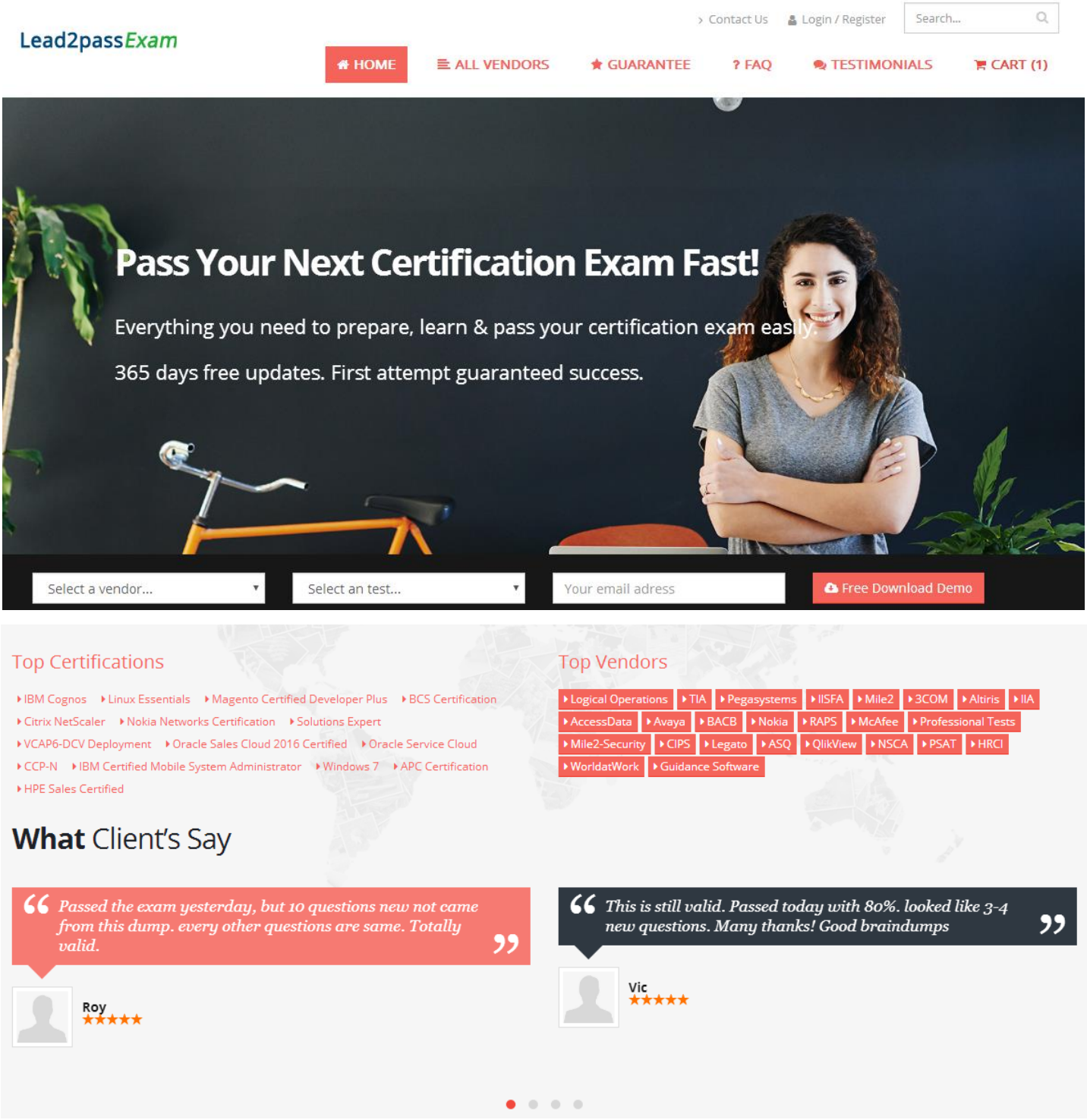
Lead2passExam



http://www.lead2passexam.com/

Available Exam Cram and Valid Dumps - Lead2Pass Exam

Exam : **70-354**

Title : Universal Windows Platform - App Architecture and UX/UI (beta)

Vendor : Microsoft

Version : DEMO

NO.1 You are packaging a Universal Windows Platform (UWP) app for the Microsoft Store.
You need to set the name of the app that will appear in the Store.
Which element should you define in Package.appxmanifest?
A. the Name attribute in the Identity element
B. the Application element
C. the DisplayName attribute in the VisualElements element
D. the DisplayName element
Answer: C

NO.2 You are developing a Universal Windows Platform (UWP) app that will be published to the Microsoft Store.
You need to ensure that the app can access removable storage drives.
Which file should you modify?
A. Package.appxmanifest . Project.json . Project.csproj
B. App.xaml.cs
Answer: A

NO.3 You are developing a Universal Windows Platform (UWP) app by using Microsoft Visual Studio 2015.
You plan to perform unit testing.
You need to automate the creation of the unit tests.
What should you use?
A. IntelliTest
B. Application Insights
C. Code Coverage
D. Test Explorer
Answer: A

NO.4 You need to provide license metadata to the current app. Which class should you use?
A. ListingInformation
B. LicenseInformation
C. ProductListing
D. ProductLicense
Answer: C

NO.5 HOTSPOT
You have the following code:
eventl += new PointerEventHandler(Target_PointerPressed); event2 += new PointerEventHandler(Target_PointerWheelChange); event3 += new PointerEventHandler(Target_PointerReleased);
For each of the following statements, select Yes if the statement is true. Otherwise, select No.

Statement	Yes	No
Event1 supports touch gestures.	O	O
Event2 supports touch gestures.	O	O
Event3 supports touch gestures	O	O

Answer:

Statement	Yes	No
Event1 supports touch gestures.	O	O
Event2 supports touch gestures.	O	O
Event3 supports touch gestures	O	O

NO.6 DRAG DROP

You are developing a Universal Windows Platform (UWP) app that will have ink capabilities.

You have the following code that configures the InkCanvas control.

```
<Grid>
<Grid.RowDefinitions>
     <RowDefinition Height"Auto" />
      <RowDefinition Height<>="Auto" />
      <RowDefinition Height="Auto" />
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" />
      <ColumnDefinition Width="Auto" />
  </Grid.ColumnDefinitions>
  <TextBlock Text="Enter Date:" />
<TextBox x:Name="DateInput" Grid.Column="1" />
<ComboBox x:Name="CultureInput" Grid.Row="1" Grid.Column="1">
    <ComboBoxItem>en-US</ComboBoxItem>
      <ComboBoxItem>fr-FR</ComboBoxItem>
      <ComboBoxItem>fr-CA</ComboBoxItem>
      <ComboBoxItem>de-DE</ComboBoxItem>
  </ComboBox>
  <TextBlock Text="Select culture setting:" Grid.Row="1" />
<Button Content="Submit" Click="Button_Click" Grid.Row="2" />
  <TextBlock x:Name="DateOutput" Grid.Row="2" Grid.Column="1" />
</Grid>
```

You need to develop the code for Button_Click. You write the following code.

```
01.  private void Button_Click(object sender, RoutedEventArgs e)
02.  {
03.
04.  }
```

Which code should you insert at line 03? Develop the solution by selecting and arranging the required code blocks in the correct order.

NOTE: You will not need all of the code blocks.

Code Blocks

```
DateOutput.Text = $"{inputDate.ToUniversalTime
()}";
```

```
DateTime inputDate;
result = DateTime.TryParse(DateInput.Text,
  culture.DateTimeFormat, styles,
out inputDate);
```

```
DateTime inputDate = DateTime.Parse
(DateInput.Text, culture.DateTimeFormat,
styles);
```

```
if (!result)
{
DateOutput.Text = "Unable to recognize";
return;
}
```

```
bool result = false;
var culture = new CultureInfo
(CultureInput.SelectedValuePath);
DateTimeStyles styles = DateTimeStyles.AllowInnerWhite
| DateTimeStyles.AllowLeadingWhite
| DateTimeStyles.AllowTrailingWhite;
```

```
if (result)
{
DateOutput.Text = "Unable to recognize";
return;
}
```

Answer:

Code Blocks

```
DateOutput.Text = $"{inputDate.ToUniversalTime
()}";
```

```
DateTime inputDate;
result = DateTime.TryParse(DateInput.Text,
  culture.DateTimeFormat, styles,
out inputDate);
```

```
DateTime inputDate = DateTime.Parse
(DateInput.Text, culture.DateTimeFormat,
styles);
```

```
if (!result)
{
DateOutput.Text = "Unable to recognize";
return;
}
```

```
bool result = false;
var culture = new CultureInfo
(CultureInput.SelectedValuePath);
DateTimeStyles styles = DateTimeStyles.AllowInnerWhite
| DateTimeStyles.AllowLeadingWhite
| DateTimeStyles.AllowTrailingWhite;
```

```
if (result)
{
DateOutput.Text = "Unable to recognize";
return;
}
```

```
bool result = false;
var culture = new CultureInfo
(CultureInput.SelectedValuePath);
DateTimeStyles styles = DateTimeStyles.AllowInnerWhite
| DateTimeStyles.AllowLeadingWhite
| DateTimeStyles.AllowTrailingWhite;
```

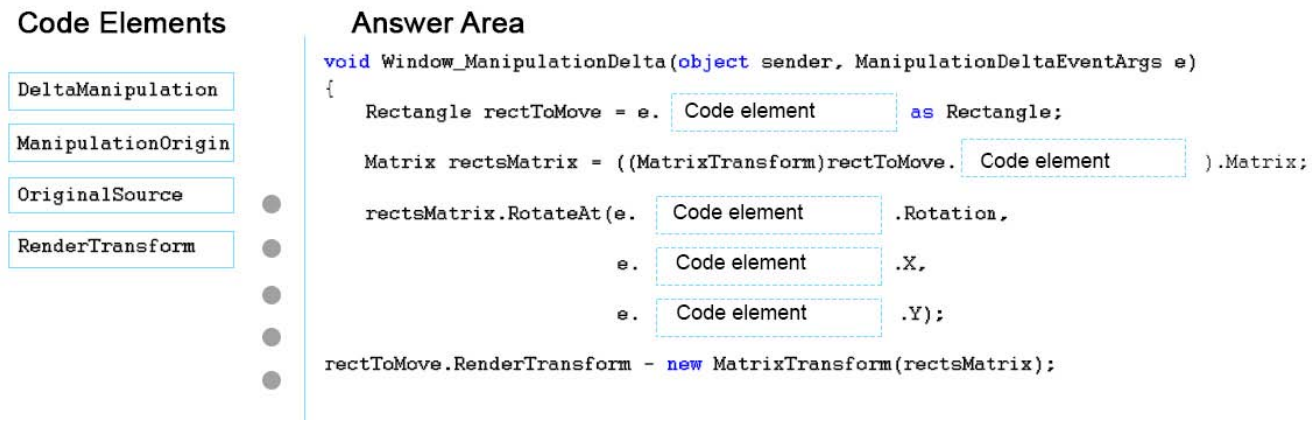
```
DateTime inputDate;
result = DateTime.TryParse(DateInput.Text,
  culture.DateTimeFormat, styles,
out inputDate);
```

```
DateOutput.Text = $"{inputDate.ToUniversalTime
()}";
```

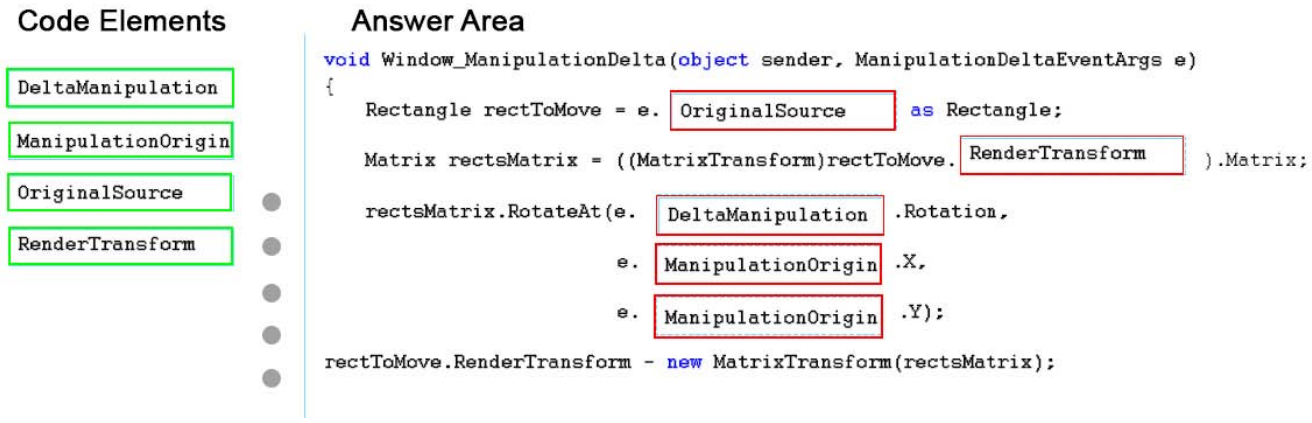
NO.7 DRAG DROP

On the details page, you render the picture of the computer inside an object of the Rectangle type. You need to implement the ability to rotate the rectangle that contains the picture.

How should you complete the code? To answer, drag the appropriate code elements to the correct targets. Each element may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.


Code Elements
DeltaManipulation
ManipulationOrigin
OriginalSource
RenderTransform
Answer Area
void Window_ManipulationDelta(object sender, ManipulationDeltaEventArgs e)
{
Rectangle rectToMove = e. Code element as Rectangle;
Matrix rectsMatrix = ((MatrixTransform)rectToMove. Code element).Matrix;
rectsMatrix.RotateAt(e. Code element .Rotation,
e. Code element .X,
e. Code element .Y);
rectToMove.RenderTransform - new MatrixTransform(rectsMatrix);

Answer:


Code Elements
DeltaManipulation
ManipulationOrigin
OriginalSource
RenderTransform
Answer Area
void Window_ManipulationDelta(object sender, ManipulationDeltaEventArgs e)
{
Rectangle rectToMove = e. OriginalSource as Rectangle;
Matrix rectsMatrix = ((MatrixTransform)rectToMove. RenderTransform).Matrix;
rectsMatrix.RotateAt(e. DeltaManipulation .Rotation,
e. ManipulationOrigin .X,
e. ManipulationOrigin .Y);
rectToMove.RenderTransform - new MatrixTransform(rectsMatrix);