

1,a

- a. Write an xml code to design a datagrid as shown in figure below, which consists of two text columns and two template columns. Where "Edit" and "Delete" in each row are the buttons in template column.

Name	Age	Update	Delete
danish	20	Edit	Delete
ali	20	Edit	Delete
frag	20	Edit	Delete
zayen	20	Edit	Delete

```
<Window x:Class="DataGridExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="DataGrid Example" Height="350" Width="500">
    <Grid>
        <DataGrid x:Name="dataGrid" AutoGenerateColumns="False" HeadersVisibility="Column" HorizontalScrollBarVisibility="Auto" VerticalScrollBarVisibility="Auto">
            <DataGrid.Columns>
                <!-- Text Column: Name -->
                <DataGridTextColumn Header="Name" Binding="{Binding Name}" Width="*" />

                <!-- Text Column: Age -->
                <DataGridTextColumn Header="Age" Binding="{Binding Age}" Width="*" />

                <!-- Template Column: Edit Button -->
                <DataGridTemplateColumn Header="Update">
                    <DataGridTemplateColumn.CellTemplate>
                        <DataTemplate>
                            <Button Content="Edit" Width="50" Click="EditButton_Click" />
                        </DataTemplate>
                    </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>

                <!-- Template Column: Delete Button -->
                <DataGridTemplateColumn Header="Delete">
                    <DataGridTemplateColumn.CellTemplate>
                        <DataTemplate>
                            <Button Content="Delete" Width="50" Click="DeleteButton_Click" />
                        </DataTemplate>
                    </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>
            </DataGrid.Columns>
        </DataGrid>
    </Grid>
</Window>
```

<Window x:Class="DataGridExample.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

```

Title="DataGrid Example" Height="350" Width="500">

<Grid>

    <DataGrid x:Name="dataGrid" AutoGenerateColumns="False"
HeadersVisibility="Column" HorizontalAlignment="Center" VerticalAlignment="Center"
Width="450">

        <DataGrid.Columns>

            <!-- Text Column: Name -->

            <DataGridTextColumn Header="Name" Binding="{Binding Name}" Width="*" />


            <!-- Text Column: Age -->

            <DataGridTextColumn Header="Age" Binding="{Binding Age}" Width="*" />


            <!-- Template Column: Edit Button -->

            <DataGridTemplateColumn Header="Update">

                <DataGridTemplateColumn.CellTemplate>

                    <DataTemplate>

                        <Button Content="Edit" Width="50" Click="EditButton_Click" />

                    </DataTemplate>

                </DataGridTemplateColumn.CellTemplate>

            </DataGridTemplateColumn>


            <!-- Template Column: Delete Button -->

            <DataGridTemplateColumn Header="Delete">

                <DataGridTemplateColumn.CellTemplate>

                    <DataTemplate>

                        <Button Content="Delete" Width="50" Click="DeleteButton_Click" />

                    </DataTemplate>

                </DataGridTemplateColumn.CellTemplate>

            </DataGridTemplateColumn>

        </DataGrid.Columns>

    </DataGrid>

</Grid>

```

</Window>

## Optional(1a)

```
using System.Collections.Generic;
```

```
using System.Windows;
```

```
using System.Windows.Controls;
```

```
namespace DataGridExample
```

```
{
```

```
    public partial class MainWindow : Window
```

```
    {
```

```
        public class Person
```

```
        {
```

```
            public string Name { get; set; }
```

```
            public int Age { get; set; }
```

```
        }
```

```
        public MainWindow()
```

```
        {
```

```
            InitializeComponent();
```

```
            LoadData();
```

```
        }
```

```
        private void LoadData()
```

```
        {
```

```
            List<Person> people = new List<Person>
```

```
            {
```

```
                new Person { Name = "danish", Age = 20 },
```

```
                new Person { Name = "ali", Age = 20 },
```

```
                new Person { Name = "fraz", Age = 20 },
```

```
                new Person { Name = "zayan", Age = 20 }
```

```
            };
```

```

        dataGrid.ItemsSource = people;
    }

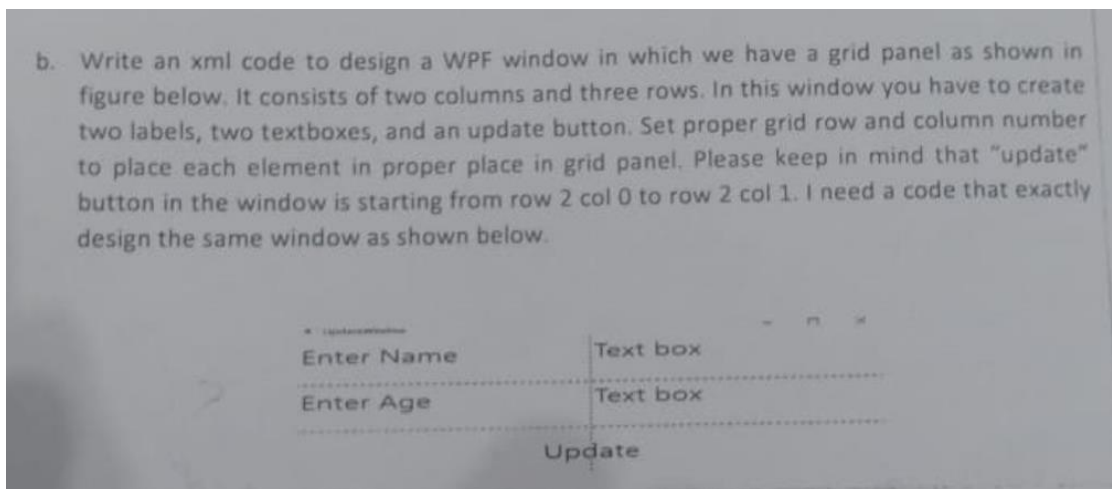
    private void EditButton_Click(object sender, RoutedEventArgs e)
    {
        MessageBox.Show("Edit button clicked!");
    }

    private void DeleteButton_Click(object sender, RoutedEventArgs e)
    {
        MessageBox.Show("Delete button clicked!");
    }
}
}

```

1,b

b. Write an xml code to design a WPF window in which we have a grid panel as shown in figure below. It consists of two columns and three rows. In this window you have to create two labels, two textboxes, and an update button. Set proper grid row and column number to place each element in proper place in grid panel. Please keep in mind that "update" button in the window is starting from row 2 col 0 to row 2 col 1. I need a code that exactly design the same window as shown below.



xml

Copy Edit

```
<Window x:Class="GridPanelExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Grid Panel Example" Height="250" Width="300">

    <Grid>

        <!-- Define 3 Rows and 2 Columns -->
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="1*" />
        </Grid.ColumnDefinitions>

        <!-- Label: Enter Name -->
        <TextBlock Text="Enter Name" Grid.Row="0" Grid.Column="0" VerticalAlignment="Center" />

        <!-- TextBox for Name -->
        <TextBox Grid.Row="0" Grid.Column="1" Margin="10"/>

        <!-- Label: Enter Age -->
        <TextBlock Text="Enter Age" Grid.Row="1" Grid.Column="0" VerticalAlignment="Center" />

        <!-- TextBox for Age -->
        <TextBox Grid.Row="1" Grid.Column="1" Margin="10"/>

        <!-- Update Button (Spanning Across 2 Columns) -->
        <Button Content="Update" Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="2"
                HorizontalAlignment="Center" Margin="10" Width="80"/>

    </Grid>
</Window>
```

2

Develop a C# code in which you have to do the following tasks

- a. Prepare a class "Person" with two properties "Name" and "Age". Implement "INotifyPropertyChanged" interface and call a notify function whenever the property "Name" or "Age" changed.
- b. Prepare a class "DataHandler" in which you have to declare observablecollections of type person and write delete, update and getalldata functions. Getalldata function will return observablecollections list, Delete function will Delete the selected object from list and update will change the updated values.
- c. In mainwindow.xml.cs file perform databinding using datacontext between the observable list and datagrid already designed in Q1 part a.
- d. Write a click\_on\_delete\_button event in mainwindow.xml.cs file and then delete the selected item from the observablecollection.
- e. Write a click\_on\_edit\_button event in mainwindow.xml.cs file and write an appropriate code to open new window already designed in Q1 part b. Take updated values and update it in observablelist.

## 1. Create the `Person` class implementing `INotifyPropertyChanged`

csharp

Copy Edit

```
using System.ComponentModel;

public class Person : INotifyPropertyChanged
{
    private string _name;
    private int _age;

    public string Name
    {
        get { return _name; }
        set
        {
            if (_name != value)
            {
                _name = value;
                OnPropertyChanged("Name");
            }
        }
    }

    public int Age
    {
        get { return _age; }
        set
        {
            if (_age != value)
            {
                _age = value;
                OnPropertyChanged("Age");
            }
        }
    }

    public event PropertyChangedEventHandler PropertyChanged;

    protected void OnPropertyChanged(string propertyName)
    {
        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
    }
}
```



## 2. Create the `DataHandler` class to manage the `ObservableCollection<Person>`

csharp

Copy Edit

```
using System.Collections.ObjectModel;
using System.Linq;

public class DataHandler
{
    public ObservableCollection<Person> People { get; set; }

    public DataHandler()
    {
        People = new ObservableCollection<Person>
        {
            new Person { Name = "Danish", Age = 20 },
            new Person { Name = "Ali", Age = 20 },
            new Person { Name = "Fraz", Age = 20 },
            new Person { Name = "Zayan", Age = 20 }
        };
    }

    public ObservableCollection<Person> GetAllData()
    {
        return People;
    }

    public void DeletePerson(Person person)
    {
        if (People.Contains(person))
            People.Remove(person);
    }

    public void UpdatePerson(Person oldPerson, string newName, int newAge)
    {
        var person = People.FirstOrDefault(p => p == oldPerson);
        if (person != null)
        {
            person.Name = newName;
            person.Age = newAge;
        }
    }
}
```





### 3. Bind `DataGrid` in `MainWindow.xaml`

```
xml Copy Edit

<Window x:Class="DataGridExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Person List" Height="300" Width="400">
    <Grid>
        <DataGrid x:Name="dataGrid" ItemsSource="{Binding People}" AutoGenerateColumns="False"
                  HorizontalAlignment="Center" VerticalAlignment="Center" Width="350">
            <DataGrid.Columns>
                <DataGridTextColumn Header="Name" Binding="{Binding Name}" Width="**"/>
                <DataGridTextColumn Header="Age" Binding="{Binding Age}" Width="**"/>
                <DataGridTemplateColumn Header="Update">
                    <DataGridTemplateColumn.CellTemplate>
                        <DataTemplate>
                            <Button Content="Edit" Width="50" Click="EditButton_Click"/>
                        </DataTemplate>
                    </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>
                <DataGridTemplateColumn Header="Delete">
                    <DataGridTemplateColumn.CellTemplate>
                        <DataTemplate>
                            <Button Content="Delete" Width="50" Click="DeleteButton_Click"/>
                        </DataTemplate>
                    </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>
            </DataGrid.Columns>
        </DataGrid>
    </Grid>
</Window>
```

## 4. Implement `MainWindow.xaml.cs`

csharp

Copy Edit

```
using System.Windows;
using System.Windows.Controls;

namespace DataGridExample
{
    public partial class MainWindow : Window
    {
        private DataHandler dataHandler;

        public MainWindow()
        {
            InitializeComponent();
            dataHandler = new DataHandler();
            DataContext = dataHandler;
        }

        private void DeleteButton_Click(object sender, RoutedEventArgs e)
        {
            if (dataGrid.SelectedItem is Person selectedPerson)
            {
                dataHandler.DeletePerson(selectedPerson);
            }
        }

        private void EditButton_Click(object sender, RoutedEventArgs e)
        {
            if (dataGrid.SelectedItem is Person selectedPerson)
            {
                EditWindow editWindow = new EditWindow(selectedPerson);
                editWindow.ShowDialog();
                dataGrid.Items.Refresh();
            }
        }
    }
}
```

## 5. Create `EditWindow.xaml` for Editing Person Details

```
xml                                                                    Copy Edit

<Window x:Class="DataGridExample.EditWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Edit Person" Height="200" Width="300">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="1*" />
            <ColumnDefinition Width="2*" />
        </Grid.ColumnDefinitions>

        <TextBlock Text="Name:" Grid.Row="0" Grid.Column="0" VerticalAlignment="Center" Margin="10"/>
        <TextBox x:Name="nameBox" Grid.Row="0" Grid.Column="1" Margin="10"/>

        <TextBlock Text="Age:" Grid.Row="1" Grid.Column="0" VerticalAlignment="Center" Margin="10"/>
        <TextBox x:Name="ageBox" Grid.Row="1" Grid.Column="1" Margin="10"/>

        <Button Content="Save" Grid.Row="2" Grid.Column="0" Grid.ColumnSpan="2"
                HorizontalAlignment="Center" Margin="10" Click="SaveButton_Click"/>
    </Grid>
</Window>
```

## 6. Implement `EditWindow.xaml.cs`

```
csharp Copy Edit

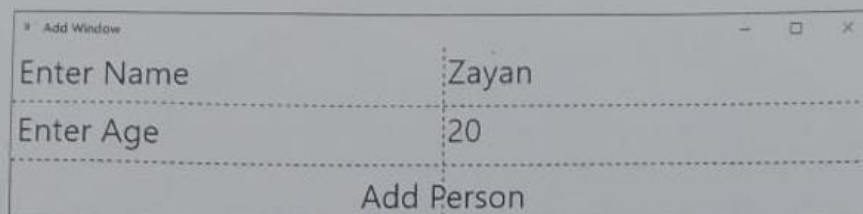
using System;
using System.Windows;

namespace DataGridExample
{
    public partial class EditWindow : Window
    {
        private Person person;

        public EditWindow(Person selectedPerson)
        {
            InitializeComponent();
            person = selectedPerson;
            nameBox.Text = person.Name;
            ageBox.Text = person.Age.ToString();
        }

        private void SaveButton_Click(object sender, RoutedEventArgs e)
        {
            if (!string.IsNullOrEmpty(nameBox.Text) && int.TryParse(ageBox.Text, out int newAge))
            {
                person.Name = nameBox.Text;
                person.Age = newAge;
                Close();
            }
            else
            {
                MessageBox.Show("Invalid input! Please enter valid details.");
            }
        }
    }
}
```

Consider the following "Add Window". You don't need to write its xml design code, just write a click event function of "Add Person" button in which you have to insert "Name" and "Age" of a person in a database table "dbo.person". I need proper code in which you have to create connections, strings, commands etc. Also write such a code that avoid SQL attacks from client sides. Consider the connection string is "database();mydatabase(Local)".



Enter Name	Zayan
Enter Age	20
Add Person	

```

using System;
using System.Data.SqlClient;
using System.Windows;

namespace AddPersonApp
{
    public partial class Addwindow : Window
    {
        private string connectionString = "database():mydatabase(Local)";

        public Addwindow()
        {
            InitializeComponent();
        }

        private void AddPerson_Click(object sender, RoutedEventArgs e)
        {
            string name = nameTextBox.Text.Trim();
            string ageText = ageTextBox.Text.Trim();

            if (string.IsNullOrEmpty(name) || string.IsNullOrEmpty(ageText) || !int.TryParse(ageText, out int age))
            {
                MessageBox.Show("Invalid input! Please enter a valid name and age.", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                return;
            }

            try
            {
                using (SqlConnection connection = new SqlConnection(connectionString))
                {
                    connection.Open();

                    string query = "INSERT INTO dbo.person (Name, Age) VALUES (@Name, @Age)";

32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
                    string query = "INSERT INTO dbo.person (Name, Age) VALUES (@Name, @Age)";

                    using (SqlCommand command = new SqlCommand(query, connection))
                    {
                        command.Parameters.AddWithValue("@Name", name);
                        command.Parameters.AddWithValue("@Age", age);

                        int rowsAffected = command.ExecuteNonQuery();

                        if (rowsAffected > 0)
                        {
                            MessageBox.Show("Person added successfully!", "Success", MessageBoxButton.OK, MessageBoxImage.Information);
                            nameTextBox.Clear();
                            ageTextBox.Clear();
                        }
                        else
                        {
                            MessageBox.Show("Failed to add person.", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
                        }
                    }
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Database Error: {ex.Message}", "Error", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }
}

```

Compute the output of the following code. If you feel any error in the code then please highlight the line.

```
namespace mynamespace
{
    delegate void delegate1();
    delegate void delegate2(string num1);
    delegate void delegate3(int temp);
    class book
    {
        static void display()
        {
            Console.WriteLine("Pakistan Zindabad");
        }
        static void display(int num1)
        {
            Console.WriteLine("I love UET KSK");
        }
        static void display(string temp)
        {
            Console.WriteLine("Best of Luck");
        }
    }
    public static void Main(string[] args)
    {
        delegate1 d1 = new delegate1(display);
        delegate2 d2 = new delegate2(display);
        delegate3 d3 = new delegate3(display);
        d1.Invoke();
        d1();
    }
}
```

## Analysis of the Given C# Code

Code:

```
csharp Copy Edit

namespace mynamespace
{
    delegate void delegate1();
    delegate void delegate2(string num1);
    delegate void delegate3(int temp);

    class book
    {
        static void display()
        {
            Console.WriteLine("Pakistan Zindabad");
        }

        static void display(int num1)
        {
            Console.WriteLine("I love UET KSK");
        }

        static void display(string temp)
        {
            Console.WriteLine("Best of Luck");
        }

        public static void Main(string[] args)
        {
            delegate1 d1 = new delegate1(display);
            delegate2 d2 = new delegate2(display);
            delegate3 d3 = new delegate3(display);

            d1.Invoke();
            d1();
        }
    }
}
```

## Expected Output

The code execution should result in the following output:

nginx

Copy Edit

```
Pakistan Zindabad
Pakistan Zindabad
```

### ✓ Explanation:

- `delegate1 d1 = new delegate1(display);`
  - Binds `d1` to the **parameterless** `display()` method.
- `d1.Invoke();` and `d1();`
  - Both lines execute the **parameterless** `display()` method, which prints `"Pakistan Zindabad"` **twice**.

## Errors in the Code

### ✗ Errors in Delegate Assignments for `d2` and `d3`

csharp

Copy Edit

```
delegate2 d2 = new delegate2(display);
delegate3 d3 = new delegate3(display);
```

### Issue:

- `delegate2` expects a **string** parameter but `display` is overloaded (has multiple versions).
- `delegate3` expects an **int** parameter, but the compiler does not automatically resolve which `display()` method to use.
- In **C#**, **method overload resolution does not work with delegate assignment like this**.
- The compiler cannot distinguish which `display(int)` or `display(string)` function to bind to `d2` and `d3`.





Fix: Explicitly specify which method to assign:

csharp

Copy Edit

```
delegate2 d2 = new delegate2(display);  
delegate3 d3 = new delegate3(display);
```

should be:

csharp

Copy Edit

```
delegate2 d2 = new delegate2(display);  
delegate3 d3 = new delegate3(display);
```

should be replaced with:

csharp

Copy Edit

```
delegate2 d2 = new delegate2(display); // ✗ Error: Overload resolution issue  
delegate3 d3 = new delegate3(display); // ✗ Error: Overload resolution issue
```

Correct version:

csharp

Copy Edit

```
delegate2 d2 = new delegate2(display); // ✗ Error: Overload resolution issue  
delegate3 d3 = new delegate3(display); // ✗ Error: Overload resolution issue
```

Corrected version:

csharp

Copy Edit

```
delegate2 d2 = new delegate2(display);  
delegate3 d3 = new delegate3(display);
```