**Given Tasks**

**First Task:**

We have a class representing binary tree nodes: class BTN { int val; BTN left; BTN right; } please implement method to compare 2 such trees (returns: true if fully equal, false - otherwise)

**Ans:**

[BTN.cs](https://github.com/varshanihanth/Tasks_L/blob/master/FirstTask/FirstTask/BTN.cs):

|  |
| --- |
|  |
|  | using System;  using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace FirstTask |
|  | { |
|  | public class BTN |
|  | { |
|  | public int val; |
|  | public BTN left; |
|  | public BTN right; |
|  | } |
|  | } |
| [Program.cs](https://github.com/varshanihanth/Tasks_L/blob/master/FirstTask/FirstTask/Program.cs): |
|  | using System;  using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace FirstTask |
|  | { |
|  | class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | // create first BTN data |
|  | var firstBTN = new BTN |
|  | { |
|  | val = 1, |
|  | right = new BTN |
|  | { |
|  | val = 2, |
|  | right = new BTN |
|  | { |
|  | val = 3 |
|  | } |
|  | } |
|  | }; |
|  |  |
|  | // create second BTN data |
|  | var secondBTN = new BTN |
|  | { |
|  | val = 1, |
|  | right = new BTN |
|  | { |
|  | val = 2, |
|  | right = new BTN |
|  | { |
|  | val = 3 |
|  | } |
|  | } |
|  | }; |
|  |  |
|  | // compare btn |
|  | bool isEqual = CompareBTN(firstBTN, secondBTN); |
|  |  |
|  | //console the result |
|  | Console.WriteLine("Is Equal: " + isEqual); |
|  | Console.ReadLine(); |
|  | } |
|  |  |
|  | public static bool CompareBTN(BTN firstBTN, BTN secondBTN) |
|  | { |
|  | //base case : check both are null |
|  | if (firstBTN == null && secondBTN == null) |
|  | return true; |
|  | //base case : check any of these is null |
|  | if (firstBTN == null || secondBTN == null) |
|  | return false; |
|  |  |
|  | if (firstBTN.val == secondBTN.val) |
|  | { |
|  | //subtrees |
|  | bool left = CompareBTN(firstBTN.left, secondBTN.left); |
|  | bool right = CompareBTN(firstBTN.right, secondBTN.right); |
|  | return (left && right); |
|  | } |
|  | else |
|  | { |
|  | return false; |
|  | } |
|  | } |
|  |  |
|  | } |
|  | } |

**Second Task:**

We have a class : class Element { int num; String name; int age; } please implement method taking Collection as an argument and returning Collection where all elements are unique by "num" and age>20. In case of ambiguity, put in target collection any of input elements.

**Ans:**

[Element.cs](https://github.com/varshanihanth/Tasks_L/blob/master/SecondTask/SecondTask/Element.cs)

|  |
| --- |
|  |
|  | using System;  using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace SecondTask |
|  | { |
|  | public class Element |
|  | { |
|  | public int Num { get; set; } |
|  | public string Name { get; set; } |
|  | public int Age { get; set; } |
|  | } |
|  | } |

[Program.cs](https://github.com/varshanihanth/Tasks_L/blob/master/SecondTask/SecondTask/Program.cs)

|  |
| --- |
|  |
|  | using Newtonsoft.Json;  using System; |
|  | using System.Collections.Generic; |
|  | using System.Linq; |
|  | using System.Text; |
|  | using System.Threading.Tasks; |
|  |  |
|  | namespace SecondTask |
|  | { |
|  | class Program |
|  | { |
|  | static void Main(string[] args) |
|  | { |
|  | // Create a list for data |
|  | var elementCollection = new List<Element> |
|  | { |
|  | new Element |
|  | { |
|  | Num=1, |
|  | Name= "Nihanth", |
|  | Age=25 |
|  | }, |
|  | new Element |
|  | { |
|  | Num=3, |
|  | Name= "Varsha", |
|  | Age=22 |
|  | }, |
|  | new Element |
|  | { |
|  | Num=2, |
|  | Name= "Reddy", |
|  | Age=22 |
|  | } |
|  | }; |
|  |  |
|  |  |
|  | //get unique elements |
|  | var uniqueElements = GetUnique(elementCollection); |
|  |  |
|  | // convert List to the string json for console.Writeline |
|  | var jsonElements = JsonConvert.SerializeObject(uniqueElements); |
|  | Console.WriteLine(jsonElements); |
|  |  |
|  | Console.ReadLine(); |
|  | } |
|  |  |
|  | /// <summary> |
|  | /// Method takes List<Element> as an argument and returns List<Element> where all elements are unique by "num" and age > 20. |
|  | /// </summary> |
|  | /// <param name="elements">ICollection<Element> </param> |
|  | /// <returns>ICollection<Element></returns> |
|  | private static List<Element> GetUnique(List<Element> elements) |
|  | { |
|  | List<Element> returnedElements = elements.Where(e => e.Age > 20) |
|  | .GroupBy(g => new |
|  | { |
|  | g.Num |
|  | }) |
|  | .Select(s => new Element |
|  | { |
|  | Num = s.Key.Num, |
|  | Name = s.FirstOrDefault().Name, |
|  | Age = s.FirstOrDefault().Age |
|  |  |
|  | }).ToList(); |
|  |  |
|  | return returnedElements; |
|  | } |
|  | } |
|  | } |

**Third Task:**

We do have 2 tables in a relational database: ∙ Clients (client\_id(PK), client\_name) ∙ Orders (order\_id(PK), client\_id, order\_sum, order\_date) Please write the following SQL queries:

a- list of clients, which have an order with order\_sum > 50

b- clients, whose total sum of orders exceeds 100.

**Ans:**

[DBscript.sql](https://github.com/varshanihanth/Tasks_L/blob/master/ThirdTask/DBscript.sql)

|  |
| --- |
|  |
|  |  |
|  | /\*\*\*\*\*\* Object: Table [dbo].[Clients] Script Date: 06/14/2018 5:15:18 PM \*\*\*\*\*\*/  CREATE TABLE [dbo].[Clients]( |
|  | [client\_id] [int] IDENTITY(1,1) NOT NULL, |
|  | [client\_name] [varchar](max) NULL, |
|  | PRIMARY KEY CLUSTERED |
|  | ( |
|  | [client\_id] ASC |
|  | )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY] |
|  | ) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY] |
|  | GO |
|  | /\*\*\*\*\*\* Object: Table [dbo].[Orders] Script Date: 06/14/2018 5:18:18 PM \*\*\*\*\*\*/ |
|  | SET ANSI\_NULLS ON |
|  | GO |
|  | SET QUOTED\_IDENTIFIER ON |
|  | GO |
|  | CREATE TABLE [dbo].[Orders]( |
|  | [order\_id] [int] IDENTITY(1,1) NOT NULL, |
|  | [client\_id] [int] NULL, |
|  | [order\_sum] [decimal](18, 0) NULL, |
|  | [order\_date] [datetime] NULL, |
|  | PRIMARY KEY CLUSTERED |
|  | ( |
|  | [order\_id] ASC |
|  | )WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY] |
|  | ) ON [PRIMARY] |
|  | GO |
|  | SET IDENTITY\_INSERT [dbo].[Clients] ON |
|  |  |
|  | INSERT [dbo].[Clients] ([client\_id], [client\_name]) VALUES (1, N'Varsha') |
|  | INSERT [dbo].[Clients] ([client\_id], [client\_name]) VALUES (2, N'Nihanth') |
|  | INSERT [dbo].[Clients] ([client\_id], [client\_name]) VALUES (3, N'Reddy') |
|  | INSERT [dbo].[Clients] ([client\_id], [client\_name]) VALUES (4, N'Vangala') |
|  | SET IDENTITY\_INSERT [dbo].[Clients] OFF |
|  |  |
|  | SET IDENTITY\_INSERT [dbo].[Orders] ON |
|  | INSERT [dbo].[Orders] ([order\_id], [client\_id], [order\_sum], [order\_date]) VALUES (1, 1, CAST(55 AS Decimal(18, 0)), CAST(N'2018-06-14T15:37:18.000' AS DateTime)) |
|  | INSERT [dbo].[Orders] ([order\_id], [client\_id], [order\_sum], [order\_date]) VALUES (2, 3, CAST(56 AS Decimal(18, 0)), CAST(N'2018-06-14T15:37:53.000' AS DateTime)) |
|  | INSERT [dbo].[Orders] ([order\_id], [client\_id], [order\_sum], [order\_date]) VALUES (3, 2, CAST(531 AS Decimal(18, 0)), CAST(N'2018-06-14T15:38:08.000' AS DateTime)) |
|  | INSERT [dbo].[Orders] ([order\_id], [client\_id], [order\_sum], [order\_date]) VALUES (4, 4, CAST(67 AS Decimal(18, 0)), CAST(N'2018-06-14T15:38:18.000' AS DateTime)) |
|  | INSERT [dbo].[Orders] ([order\_id], [client\_id], [order\_sum], [order\_date]) VALUES (5, 1, CAST(52 AS Decimal(18, 0)), CAST(N'2018-06-14T15:38:49.000' AS DateTime)) |
|  | SET IDENTITY\_INSERT [dbo].[Orders] OFF |
|  |  |
|  | ALTER TABLE [dbo].[Orders] WITH CHECK ADD CONSTRAINT [order\_client\_id] FOREIGN KEY([client\_id]) |
|  | REFERENCES [dbo].[Clients] ([client\_id]) |
|  | ON DELETE CASCADE |
|  | GO |
|  | ALTER TABLE [dbo].[Orders] CHECK CONSTRAINT [order\_client\_id] |

[Query A.txt](https://github.com/varshanihanth/Tasks_L/blob/master/ThirdTask/Query%20A.txt)

|  |
| --- |
|  |
|  | SELECT \* from Clients c  INNER JOIN Orders o on c.client\_id= o.client\_id |
|  | WHERE o.order\_sum > 50 |
|  |  |
|  | SELECT c.client\_id, c.client\_name, SUM(o.order\_sum) as order\_sum from Clients c |
|  | INNER JOIN Orders o on c.client\_id= o.client\_id |
|  | where o.order\_sum > 50 |
|  | GROUP BY c.client\_id ,c.client\_name  (OR)  select from clients.\* join Orders on Orders.client\_id = Clients.client\_id where Orders.order\_sum > 50 |

[Query B.txt](https://github.com/varshanihanth/Tasks_L/blob/master/ThirdTask/Query%20B.txt)

|  |  |  |
| --- | --- | --- |
|  | | |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | | |
|  | |  | | |
|  | |  | | |
|  | |  | | |
|  | |  | | |
|  | |  | | |
|  | |  | | |
|  | |  | | |
|  | |  | | |
|  | |  | | |
| |  | | --- | |  | |  | SELECT c.client\_id, c.client\_name, o.order\_sum from Clients c  INNER JOIN (SELECT client\_id, SUM(Orders.order\_sum) as order\_sum from Orders GROUP BY client\_id) o on c.client\_id= o.client\_id | |  | WHERE o.order\_sum > 100 | |  |  | |  |  | |  |  | |  | SELECT c.client\_id,c.client\_name , SUM(o.order\_sum ) as order\_sum from Clients c | |  | INNER JOIN Orders o on c.client\_id= o.client\_id | |  | GROUP BY c.client\_id ,c.client\_name | |  | HAVING SUM(o.order\_sum) > 100 |   (OR)  select from clients.\* join Orders on Orders.client\_id = Clients.client\_id where Orders.order\_sum > 50 | | | |