Coding test .NET

Please write a small program for each of the exercises

**1. Denomination routine**

An ATM has three cartridges for different denominations:

* 10 EUR cartridge
* 50 EUR cartridge
* 100 EUR cartridge

Now we want to pay out following amounts from the ATM:

* 30 EUR
* 50 EUR
* 60 EUR
* 80 EUR
* 140 EUR
* 230 EUR
* 370 EUR
* 610 EUR
* 980 EUR

Write a program which will calculate for each payout the possible combinations which the ATM can pay out.

For example, for 100 EUR the available payout denominations would be:

* 10 x 10 EUR
* 1 x 50 EUR + 5 x 10 EUR
* 2 x 50 EUR
* 1 x 100 EUR

A: **1. Denomination routine**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text.RegularExpressions;

namespace ePay\_Test\_ATM

{

public class Program

{

public static void Main(string[] args)

{

// initiate the value of notes type

// and empty list for combantion

List<int> combanations = new List<int>();

List<int> billType = new List<int>() { 10, 50, 100 };

// ask the user to input the amount which we can replace by an array and loop runs through the test cases

// I did not like the output of that

// though I wrote the code to work with either method.

//Console.WriteLine("How much would you like to withdraw?");

//Console.Write("Enter Amount: ");

//// to get and validate the user input

//int withdraw = GetWithdrawAmount();

//Change(combanations, billType, 0, 0, withdraw);

List<int> withdraw = new List<int>(){30,50,60,80,140,230,370,610,980};

foreach(int amount in withdraw)

{

Console.WriteLine("combinations for amount " + amount + " are:");

Change(combanations, billType, 0, 0, amount);

}

}

static int GetWithdrawAmount()

{

bool valid = false;

int withdrawAmount = Convert.ToInt32(Console.ReadLine());

while (!valid)

{

if (withdrawAmount == 0)

{

Console.WriteLine("enter correct amoount. the amount should not be less than 10€");

}

else if (withdrawAmount < 10)

{

Console.WriteLine("enter correct amoount. the amount should not be less than 10€");

}

else if (withdrawAmount % 10 != 0)

{

Console.WriteLine("enter correct amoount. the amount should not be multply by 10");

}

else

{

valid = true;

break;

}

Console.Write("Enter Amount: ");

withdrawAmount = Convert.ToInt32(Console.ReadLine());

}

return withdrawAmount;

}

static void Change(List<int> combanations, List<int> billType, int top, int total, int amount)

{

// check if are meet the withdraw amount

if (total == amount)

{

Display(combanations, billType);

return;

}

// to check if we did not exceeds the amount

if (total > amount)

{

return;

}

// Loop through amounts.

foreach (int value in billType)

{

// add higher or equal amounts.

if (value >= top)

{

List<int> copy = new List<int>(combanations);

copy.Add(value);

Change(copy, billType, value, total + value, amount);

}

}

}

static void Display(List<int> combanations, List<int> billType)

{

foreach (int bill in billType)

{

int count = combanations.Count(value => value == bill);

Console.WriteLine("{0}: {1}", bill, count);

}

Console.WriteLine();

}

}

} **2. REST server**

A small REST server with good performance for simple customer management has two functions:

* POST customers

Request:

[

{

firstName: 'Aaaa',

lastName: 'Bbbb',

age: 20,

id: 5

},

{

firstName: 'Bbbb',

lastName: 'Cccc',

age: 24,

id: 6

}

]

Multiple customers can be sent in one request.

The server validates every customer of the request:

* checks that every field is supplied
* validates that the age is above 18
* validates that the ID has not been used before

The server then adds each customer as an object to an internal array – the customers will not be appended to the array but instead it will be inserted at a position so that the customers are sorted by last name and then first name WITHOUT using any available sorting functionality (an example for the inserting is in the Appendix).

The server also persists the array so it will be still available after a restart of the server.

* GET customers

Returns the array of customers with all fields

Write the server and a small simulator which can send several requests for POST customers and GET customers in parallel to the server.

For that program it is not allowed to use any sorting mechanism like array.sort().

The simulated POST customers requests have following requirements:

* Each request should contain at least 2 different customers
* Age should be randomized between 10 and 90
* ID should be increasing sequentially.
* The first names and last names of the Appendix should be used in random combinations

**Appendix:**

**Data:**

First names: Last names:

Leia Liberty

Sadie Ray

Jose Harrison

Sara Ronan

Frank Drew

Dewey Powell

Tomas Larsen

Joel Chan

Lukas Anderson

Carlos Lane

**Example for the inserting mechanism:**

Array in server:

[

{ lastName: 'Aaaa', firstName: 'Aaaa', age: 20, id: 3 },

{ lastName: 'Aaaa', firstName: 'Bbbb', age: 56, id: 2 },

{ lastName: 'Cccc', firstName: 'Aaaa', age: 32, id: 5 },

{ lastName: 'Cccc', firstName: 'Bbbb', age: 50, id: 1 },

{ lastName: 'Dddd', firstName: 'Aaaa', age: 70, id: 4 },

]

Request POST customers:

[{ lastName: 'Bbbb', firstName: 'Bbbb', age: 26, id: 6 }]

Array after insert:

[

{ lastName: 'Aaaa', firstName: 'Aaaa', age: 20, id: 3 },

{ lastName: 'Aaaa', firstName: 'Bbbb', age: 56, id: 2 },

{ lastName: 'Bbbb', firstName: 'Bbbb', age: 26, id: 6 },

{ lastName: 'Cccc', firstName: 'Aaaa', age: 32, id: 5 },

{ lastName: 'Cccc', firstName: 'Bbbb', age: 50, id: 1 },

{ lastName: 'Dddd', firstName: 'Aaaa', age: 70, id: 4 },

]

Request POST customers:

[{ lastName: 'Bbbb', firstName: 'Aaaa', age: 28, id: 7 }]

Array after insert:

[

{ lastName: 'Aaaa', firstName: 'Aaaa', age: 20, id: 3 },

{ lastName: 'Aaaa', firstName: 'Bbbb', age: 56, id: 2 },

{ lastName: 'Bbbb', firstName: 'Aaaa', age: 28, id: 7 },

{ lastName: 'Bbbb', firstName: 'Bbbb', age: 26, id: 6 },

{ lastName: 'Cccc', firstName: 'Aaaa', age: 32, id: 5 },

{ lastName: 'Cccc', firstName: 'Bbbb', age: 50, id: 1 },

{ lastName: 'Dddd', firstName: 'Aaaa', age: 70, id: 4 },

]