**Code Test**

**Buat lah logic function dalam format coding C#.**

**Sorting dan Alokasi**

Diberikan data set sebagai berikut:

|  |  |  |
| --- | --- | --- |
| Tagihan#1 | Due: 10 Jan 23 | 165,000 |
| Tagihan#2 | Due: 15 Feb 23 | 80,000 |
| Tagihan#3 | Due: 20 Jan 23 | 130,000 |
| Tagihan#4 | Due: 25 Mar 23 | 416,000 |
| Tagihan#5 | Due: 10 Feb 23 | 95,500 |
| Tagihan#6 | Due: 17 Aug 23 | 523,000 |

Output Jawaban :

Untuk mengeluarkan list dataset diatas berikut endpoint api nya bisa di akses dengan Alamat url :

[**http://localhost:5002/Payment**](http://localhost:5002/Payment)

**A screenshot of a computer

Description automatically generated**

Berikut snapshot code

A computer screen shot of a program code

Description automatically generated

**Buatkan algoritma alokasi payment**. Input adalah nominal payment. Contoh Output adalah sebagai berikut:

1. Input Payment = 200,000

|  |  |  |  |
| --- | --- | --- | --- |
| Tagihan#1 | Due: 10 Jan 23 | 165,000 | 165,000 |
| Tagihan#3 | Due: 20 Jan 23 | 130,000 | 35,000 |
| Tagihan#5 | Due: 10 Feb 23 | 95,500 |  |
| Tagihan#2 | Due: 15 Feb 23 | 80,000 |  |
| Tagihan#4 | Due: 25 Mar 23 | 416,000 |  |

1. Input Payment = 500,000

|  |  |  |  |
| --- | --- | --- | --- |
| Tagihan#1 | Due: 10 Jan 23 | 165,000 | 165,000 |
| Tagihan#3 | Due: 20 Jan 23 | 130,000 | 130,000 |
| Tagihan#5 | Due: 10 Feb 23 | 95,500 | 95,500 |
| Tagihan#2 | Due: 15 Feb 23 | 80,000 | 80,000 |
| Tagihan#4 | Due: 25 Mar 23 | 416,000 | 29,500 |

**Berikan juga segala abnormal scenario** yang mungkin terjadi, isi dengan message box sesuai dengan scenario nya. Misal:

* input payment bernilai < 0
* apa lagi scenario yang mungkin terjadi?

Output Jawaban :

Untuk meghitung pembayaran dengan Alamat url :

A screenshot of a computer

Description automatically generated

Result Endpoint

A screenshot of a computer

Description automatically generated

Berikut Snapshot code untuk melakukan perhitungan pembayarannya :

|  |
| --- |
| public string AddPayment(decimal PaymentAllocated)   {       try       {           string message = string.Empty;           if (PaymentAllocated < 0)           {               message = "Payment less 0 ";               return message;           }           using (IDbConnection conn = common.DBConnection)           {               conn.Open();               var paymentList = conn.GetList<TbPayment>().OrderBy(t => t.DueDate).ToList();               foreach (var payment in paymentList)               {                   if (payment.Amount >= payment.PaymentAllocated)                   {                       if (PaymentAllocated >= payment.Amount)                       {                           payment.PaymentAllocated = payment.Amount;                           PaymentAllocated -= payment.Amount;                           var paymentAllocation = new TbPayment                           {                               Id = payment.Id,                               DueDate = payment.DueDate,                               Amount = payment.Amount,                               PaymentAllocated = payment.Amount,                               UpdatedOn = DateTime.Now                           };                           conn.Update(paymentAllocation);                       }                       else                       {                           payment.PaymentAllocated = PaymentAllocated;                           // Insert Payment Allocation                           var paymentAllocation = new TbPayment                           {                               Id = payment.Id,                               DueDate = payment.DueDate,                               Amount = payment.Amount,                               PaymentAllocated = PaymentAllocated,                               UpdatedOn = DateTime.Now                           };                           conn.Update(paymentAllocation);                           break;                       }                   }               }               return message;           }       }       catch (Exception ex)       {           return common.GetErrorMessage(ServiceName + "AddPayment", ex);       }   } |

**Output hasil menjalankan endpoint diatas :**

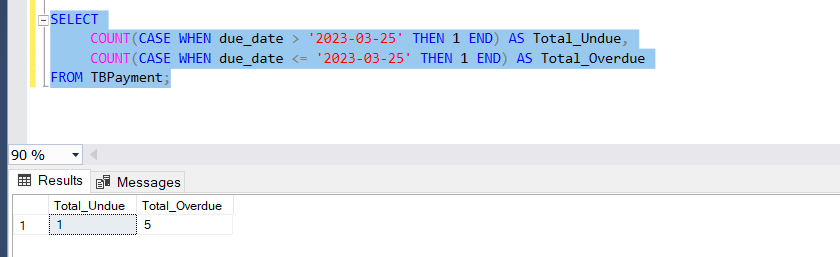
**A screenshot of a computer

Description automatically generated**

**Buat Query DB** untuk mendapatkan Total Undue (belum jatuh tempo) dan Total Overdue (sudah jatuh tempo). Dengan kondisi tanggal hari ini adalah 25 Mar 23. Dalam 1 query saja.  
  
Berikut Jawaban untuk query

|  |
| --- |
| SELECT  COUNT(CASE WHEN due\_date > '2023-03-25' THEN 1 END) AS Total\_Undue,  COUNT(CASE WHEN due\_date <= '2023-03-25' THEN 1 END) AS Total\_Overdue  FROM TBPayment; |

dan berikut output dari hasil query nya :



**Algoritma Perhitungan Penalty**

Diberikan data set sebagai berikut:

**Data Table Tagihan**

|  |  |  |
| --- | --- | --- |
| **No Tagihan** | **Due Date** | **Total Tagihan** |
| Tagihan#1 | Due: 10 Jan 23 | 165,000 |
| Tagihan#3 | Due: 20 Jan 23 | 130,000 |
| Tagihan#5 | Due: 10 Feb 23 | 95,500 |
| Tagihan#2 | Due: 15 Feb 23 | 80,000 |
| Tagihan#4 | Due: 30 Mar 23 | 416,000 |
|  |  |  |

**Data Table Pembayaran**

|  |  |  |  |
| --- | --- | --- | --- |
| **No Payment** | **No Tagihan** | **Pmt Date** | **Pmt Amount** |
| Payment#1 | Tagihan#1 | 10 Jan 23 | 165,000 |
| Payment#2 | Tagihan#3 | 20 Feb 23 | 130,000 |
| Payment#2 | Tagihan#5 | 20 Feb 23 | 95,500 |
| Payment#3 | Tagihan#2 | 25 Feb 23 | 30,000 |
| Payment#4 | Tagihan#2 | 30 Mar 23 | 50,000 |
| Payment#4 | Tagihan#4 | 30 Mar 23 | 50,000 |

Dari dataset di atas, hitunglah Penalty “keterlambatan pembayaran”.

**Rumus Penalty:** A \* 2permill \* B

A = Total Tagihan yang terlambat bayar

B = Jumlah hari keterlambatan sampai lunas

**Hasilkan dataset sebagai berikut: Hari ini = 29 Apr 22**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No Tagihan | No Penalty | Tagihan Overdue | Hari Keterlambatan | Amount Penalty |
| Tagihan#1 | 1 | 0 | 0 | 0 |
| Tagihan#3 | 1 | 130,000 | 31 | 800 |
| Tagihan#5 | 1 | 95,500 | 10 | 1,910 |
| Tagihan#2 | 1 | 30,000 | 10 | 600 |
| Tagihan#2 | 2 | 50,000 | 15 | 1,500 |
| Tagihan#4 | 1 | 366,000 | 30 | 21,960 |

Berikut Jawaban untuk menghitung nilai penalty

A screenshot of a computer

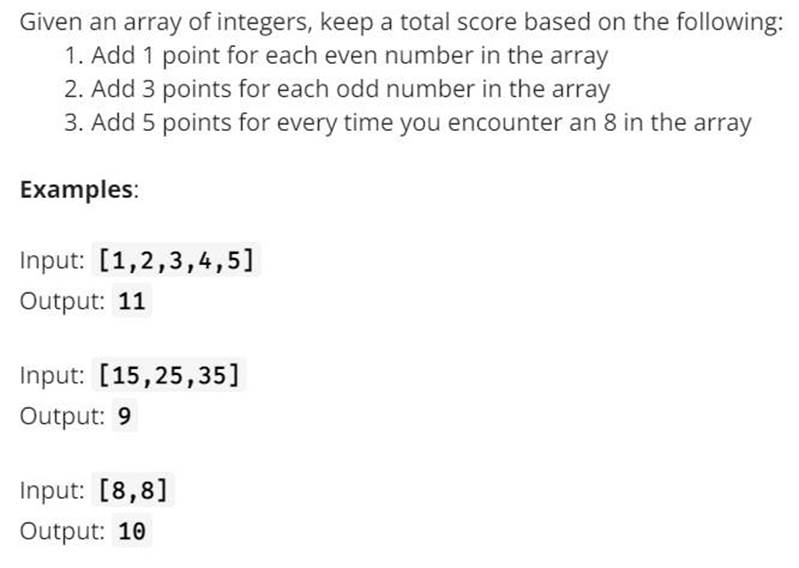
Description automatically generated

Snapshot Code

|  |
| --- |
| public List<PaymentPenaltyModel> GetPenaltyPayment(out string oMessage)  {  try  {  var today = new DateTime(2023, 04, 29);  oMessage = string.Empty;  using (IDbConnection conn = common.DBConnection)  {  conn.Open();  var payments = conn.GetList<TbPayment>().ToList();  var invoices = conn.GetList<TBPaymentInvoice>().ToList();  var penaltyResults = CalculatePenalty(invoices, payments, today);  return penaltyResults;  }  }  catch (Exception ex)  {  oMessage = common.GetErrorMessage(ServiceName + "GetPenaltyPayment", ex);  return null;  }  }   private List<PaymentPenaltyModel> CalculatePenalty(List<TBPaymentInvoice> tagihanList, List<TbPayment> pembayaranList, DateTime today)  {  var results = new List<PaymentPenaltyModel>();  // Proses setiap tagihan  foreach (var tagihan in tagihanList)  {  // Ambil daftar pembayaran untuk tagihan ini  var paymentsForTagihan = pembayaranList.Where(p => p.Id == tagihan.PaymentId).ToList();  decimal totalPaid = paymentsForTagihan.Sum(p => p.PaymentAllocated);  decimal remainingAmount = tagihan.PmtAmount - totalPaid;  if (remainingAmount > 0)  {    DateTime lastPaymentDate = paymentsForTagihan.Any() ? paymentsForTagihan.Max(p => p.DueDate) : tagihan.PmtDate;    int overdueDays = (today - lastPaymentDate).Days;  if (overdueDays < 0) overdueDays = 0; // Pastikan tidak ada nilai negatif    decimal penaltyAmount = remainingAmount \* 2m / 1000m \* overdueDays;  results.Add(new PaymentPenaltyModel  {  IdPenalty = tagihan.Id,  IdBill = tagihan.PaymentId,  BillOverDue = remainingAmount,  DayLate = overdueDays,  AmountPenalty = penaltyAmount  });  }  }  return results;  } |

**Algoritma #1**

Buatlah code untuk menghasilkan output sesuai dengan logic berikut:



**Berikut Jawaban untuk Soal Array**

|  |
| --- |
| int[] array1 = { 1, 2, 3, 4, 5 };  int[] array2 = { 15, 25, 35 };  int[] array3 = { 8, 8 };  AlgorithmOne algorithmOne = new AlgorithmOne();  Console.WriteLine($"Algorithm One: ");  Console.WriteLine("Total Score for array1: " + algorithmOne.CalculateScore(array1)); // Expected output: 11  Console.WriteLine("Total Score for array2: " + algorithmOne.CalculateScore(array2)); // Expected output: 9  Console.WriteLine("Total Score for array3: " + algorithmOne.CalculateScore(array3)); // Expected output: 10  public class AlgorithmOne  {  public int CalculateScore(int[] arr)  {  int score = 0;  foreach (int num in arr)  {  if (num == 8)  {  score += 5;  }  else if (num % 2 == 0)  {  score += 1;  }  else  {  score += 3;  }  }  return score;  }  } |

Dari hasil snap code diatas berikut ini result out put nya

A black rectangle with a white background

Description automatically generated

Berikut

**Algoritma #2**

Buatlah algoritma untuk empat soal di bawah ini, sesuai dengan nilai “n”

Contoh output di bawah adalah ketika nilai “n” = 5

Berikut jawaban snapshot code untuk soal **Algoritma #2**

|  |
| --- |
| **Output A** public void OutputA(int n)  {  for (int i = 1; i <= n; i++)  {  for (int j = 1; j <= i; j++)  {  Console.Write(i);  }  Console.WriteLine();  }  } |

**Result**

A black screen with white text

Description automatically generated

|  |
| --- |
| **Output B**  public void OutputB(int n)  {  for (int i = 1; i <= n; i++)  {  for (int j = i; j >= 1; j--)  {  Console.Write(j);  }  Console.WriteLine();  }  } |

**Result**

A black screen with white lines

Description automatically generated

|  |
| --- |
| **Output C**  public void OutputC(int n)  {  for (int i = 1; i <= n; i++)  {    for (int j = 1; j <= i; j++)  {  Console.Write(j);  }    for (int j = i - 1; j >= 1; j--)  {  Console.Write(j);  }  Console.WriteLine();  }  } |

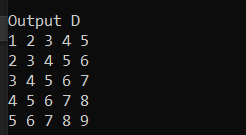
**Result**

A black screen with white text

Description automatically generated

|  |
| --- |
| Output D  public void OutputD(int n)  {  for (int i = 1; i <= n; i++)  {  for (int j = 1; j <= n; j++)  {  Console.Write((i + j - 1) + " ");  }  Console.WriteLine();  }  } |

**Result**

****

**Berikut All Resullt**

**A screenshot of a computer

Description automatically generated**

**Database Query**

Buatlah dalam 1 query untuk mendapatkan duplicate data berdasarkan 1 field.

|  |
| --- |
| Jawaban Query Duplicate  SELECT \* FROM Payments WHERE PaymentId IN ( SELECT PaymentId FROM Payments GROUP BY PaymentId HAVING COUNT(\*) > 1 ); |

Buatlah dalam 1 query antara 2 table (misal TableA dan TableB), untuk menemukan data yang missing dari TableB, berdasarkan 1 field.

**Jawabn Query Pencarian Missing 2 Table**

|  |
| --- |
| SELECT A.CustomerId, A.CustomerName FROM Customers A LEFT JOIN Orders B ON A.CustomerId = B.CustomerId WHERE B.CustomerId IS NULL; |