**Note Book - Requirement 1**

Your friend is a journalist. He finds it difficult to keep track of the notes that he is making. You being a newbie to programming want to help your friend with a small application to keep the notes in an organized way and also master programming. There are two major domains here **Note** and **Notebook**.  
  
**Requirement 1:**

Let’s start off by creating two **Note** objects and check whether they are equal.

1. Create a **Note** Class with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_name | string |
| \_content | string |
| \_size | double |
| \_priorityLevel | double |
| \_createdDate | DateTime |

1. Mark all the attributes as private
2. Include appropriate properties.
3. Add a default constructor and a parameterized constructor to take in all attributes in the given order: **Note(string \_name, string \_content, double \_size, double \_priorityLevel,DateTime \_createdDate)**
4. When the “note” object is printed, it should display the following details: **[Override the ToString method]**  
   Print format:  
   Name:"name"  
   Content:"content"  
   Size:"size"  
   Priority Level:"priorityLevel"  
   Created Date:"createdDate"
5. Two notes are considered same if they have the same \_name,\_content and \_createdDate. Implement the logic in the appropriate function. (Case – Insensitive) **[Override the Equals method]**. which returns bool to Main function(either **true** or **false**). If returns true, then print "**Note 1 is same as Note 2**", otherwise print "**Note 1 and Note 2 are different**".
6. The input format consists of note details separated by comma in the below order,  
   **(\_name,\_content,\_size,\_priorityLevel,\_createdDate**).

Create a class named as **Program**, which contains **Main**method. All the input and output operations are done in this method.  
It is used to access the above class and its method to done this requirement.  
  
The Input to your program would be details of two notes, you need to display their details as given in "5th point(refer above)" and compare the two notes and display if the Notes are same or different.  
  
**Note:**There is an empty line between display statements. Print the empty lines in main function.  
              Display one digit after the decimal point for double datatype.  
  
**Sample INPUT & OUTPUT 1:**  
  
Enter note 1 detail:  
**IPL Schedule,CSKPlayingDates,25.0,5.0,27-01-2018**

Enter note 2 detail:

**IPL Schedule,CSKPlayingDates,25.0,5.0,27-01-2018**

Note 1:

Name:IPL Schedule

Content:CSKPlayingDates

Size:25.0

Priority Level:5.0

Created Date:27-01-2018

Note 2:

Name:IPL Schedule

Content:CSKPlayingDates

Size:25.0

Priority Level:5.0

Created Date:27-01-2018

Note 1 is same as Note 2

**Sample INPUT & OUTPUT 2:**

Enter note 1 detail:

**IPL Schedule,CSKPlayingDates,25.0,5.0,27-01-2018**

Enter note 2 detail:

**Schedule,Works to be done,15.3,5.0,12-12-2018**

Note 1:

Name:IPL Schedule

Content:CSKPlayingDates

Size:25.0

Priority Level:5.0

Created Date:27-01-2018

Note 2:

Name:Schedule

Content:Works to be done

Size:15.3

Priority Level:5.0

Created Date:12-12-2018

Note 1 and Note 2 are different

**Note Book - Requirement 2**

**Requirement 2:**  
Now we are gonna start creating a notebook and add notes to it. Start with creating a notebook and use menu-driven approach to add, remove, display details of the note in the notebook.  
  
a)Create a Class **Note** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_name | string |
| \_content | string |
| \_size | double |
| \_createdDate | DateTime |
| \_priorityLevel | double |

Mark all the attributes as private.  
Include appropriate properties.  
Override **ToString()** method to display details in tabular form.  
Add a default constructor and a parameterized constructor to take in all attributes in the given order:  **public Note(string \_name, string \_content, double \_size, DateTime \_createdDate,double \_priorityLevel)**  
  
b)Create a Class **Notebook** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_name | string |
| \_noteList | List<Note> |

Mark all the attributes as private.  
Include appropriate properties.  
Add a default constructor and a parameterized constructor to take in all attributes in the given order:**Notebook(string \_name, List<Note> \_noteList).**  
In constructor pass the noteList value as an empty list. Only one notebook will be present at a time.  
  
c) Create the following static method in **Note**class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public static Note CreateNote(string detail) | This method accepts a string which contains note details separated by commas. Split the detail and create a note object from the details and return it. |

The note details should be given as a comma-separated value in the below order,  
**\_name,\_content,\_size,\_createdDate,\_priorityLevel**  
  
d) Create the following methods in **Notebook**class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public void AddNoteToBook(Note note) | This method accepts a note object and adds the note to the note list of the current notebook. |
| public bool RemoveNoteFromBook(string name) | This method will get the name of the note and delete the note with the specified name from the current notebook.  If a note with the given name found, delete the note and return **true**to Main method, and print "**Note successfully deleted**".  If a note with the name is not found return **false**to Main method and print "**Note not found in the Book**".  The book name are unique. |
| public void DisplayNotes() | This method will display the note list in the current notebook. If the note list is empty display "**No notes to show"**, else display "Notes in [notebook name]" and display all the note details in the specified format. Where [notebook name] specifies the name of the notebook. |

After deletion, if true is returned print "**Note successfully deleted**", else print "**Note not found in the Book**". After adding note to the notebook, print "**Note successfully added**".  
  
Create a class named as **Program**, which contains **Main** method. All the input and output operations are done in this method.  
It is used to access the above classes and its method to do this requirement.  
  
**Note:** The above print statements should be present in the Main method.  
  
**Problem Overview.**  
This requirement contains the menu driven,  
1.Add Note  
2.Delete Note  
3.Display Notes  
4.Exit  
  
If the user select  choice-1, then get the Note details from the user(which is comma seperated). Split that note detail and create a Note object(use CreateNote method). And add that Note to \_noteList( which is is Book ). Use AddNoteToBook method.  
If the note created and added to the noteList, then print "Note successfully added".  
  
If the user select choice-2, then get the note name from the user. Search that note from the book's note list(using note name). If the note is found and then remove the note from the note list and then print "**Note successfully deleted**". If the note is not found, then print "**Note not found in the Book**".  
  
If the user select option-3, then display the note details by following format. Refer sample input and output.

When the “note” object is printed, it should display the following format  
Print format:  
**Console.WriteLine("{0,-15}{1,-25}{2,-10}{3,-15}{4,-10}", "Name", "Content", "Size", "Created Date", "Priority Level");**  
  
**Sample Input and Output:**  
  
Enter the name of the book:  
**Party Preparation**  
1.Add Note  
2.Delete Note  
3.Display Notes  
4.Exit  
Enter your choice:  
**3**  
No notes to show  
1.Add Note  
2.Delete Note  
3.Display Notes  
4.Exit  
Enter your choice:  
**1**  
Enter the details of note in CSV format:  
**Food Items** **,Buy** **Food and bevarages** **,10,05-02-2018,3**  
Note successfully added  
1.Add Note  
2.Delete Note  
3.Display Notes  
4.Exit  
Enter your choice:  
**1**  
Enter the details of note in CSV format:  
**Decorations,Buy decoration materials,2,06-02-2018,4**  
Note successfully added  
1.Add Note  
2.Delete Note  
3.Display Notes  
4.Exit  
Enter your choice:  
**1**  
Enter the details of note in CSV format:  
**Invite,Invite Friends,15,04-02-2018,2**  
Note successfully added  
1.Add Note  
2.Delete Note  
3.Display Notes  
4.Exit  
Enter your choice:  
**3**  
Notes in :Party Preparation  
Name           Content                  Size      Created Date   Priority Level  
Food Items     Buy Food and bevarages    10.0      05-02-2018     3.0         
Decorations    Buy decoration materials 2.0       06-02-2018     4.0         
Invite         Invite Friends           15.0      04-02-2018     2.0         
1.Add Note  
2.Delete Note  
3.Display Notes  
4.Exit  
Enter your choice:  
**1**  
Enter the details of note in CSV format:  
**Alarm,Wake up on time,5,01-02-2018,1**  
Note successfully added  
1.Add Note  
2.Delete Note  
3.Display Notes  
4.Exit  
Enter your choice:  
**2**  
Enter the name of the note to be deleted:  
**alarm**  
Note not found in the Book  
1.Add Note  
2.Delete Note  
3.Display Notes  
4.Exit  
Enter your choice:  
**2**  
Enter the name of the note to be deleted:  
**Alarm**  
Note successfully deleted  
1.Add Note  
2.Delete Note  
3.Display Notes  
4.Exit  
Enter your choice:  
**3**  
Notes in :Party Preparation  
Name           Content                  Size      Created Date   Priority Level  
Food Items     Buy Food and bevarages    10.0      05-02-2018     3.0         
Decorations    Buy decoration materials 2.0       06-02-2018     4.0         
Invite         Invite Friends           15.0      04-02-2018     2.0         
1.Add Note  
2.Delete Note  
3.Display Notes  
4.Exit  
Enter your choice:  
**4**

**Note Book - Requirement 3**

**Requirement 3:**  
   In this requirement, you need to validate the name and the content of the note.  
  
a)Create a Class **Program**with the following static methods:

|  |  |  |
| --- | --- | --- |
| **Sno** | **Method name** | **Method description** |
| 1 | static bool ValidateName(string name) | Validate the name based on the rules given below. Returns **true**ifname is valid else return **false** |
| 2 | static boo ValidateContent(string content) | Validate the content based on the rules given below. Returns **true**ifcontent is valid else return **false** |

b) While validating name follow the below rules,  
   
1. The name should start only with alphabets(either uppercase or lowercase).  
2. The name should not contain any special characters.  
  
c) While validating content follow the below rules,  
  
1.The content should not contain any special characters other than**" . ", ","**and**" \_ "**.  
2.The content should not exceed the size of 40 characters excluding spaces.  
  
**Note:** Print "**Name is valid**" if name is valid else print "**Name is invalid**". Print "**Content is valid**" if email is spam else print "**Content is invalid**".  
            All the above print statements are present in the main method.  
  
That Program class also contains **Main**method. It is used to access the above methods to do this requirement.  
  
**Problem Overview:**  
This requirement contains the menu driven,  
Menu:  
1.Validate Name  
2.Validate Content  
  
If the user select option-1, then get the name of the note and validate the name( use ValidateName method). That method returns bool type (either true or false).  
If the name is validated, then returns true, then print "**Name is valid**", If the name is not valid and that method returns **false** to Main method and print "**Name is invalid**".  
  
If the user select option-2, then get the content for the note from the user and validate the content ( use  ValidateContent method). This method returns bool return type( either true or false ). If the content is validated successfully, then returns **true**to the Main function and print "**Content is valid**", If the content is not valid and that method return **false**to Main method and print "**Content is invalid**".  
  
All the input and output operations are done in **Main**method.

**Sample Input and Output 1:**  
  
1.Validate Name  
2.Validate Content  
Enter your choice:  
**1**  
Enter the name to be validated:  
**Jane1**  
Name is valid  
  
**Sample Input and Output 2:**  
  
1.Validate Name  
2.Validate Content  
Enter your choice:  
**1**  
Enter the name to be validated:  
**Jane\_1**  
Name is invalid  
  
**Sample Input and Output 3:**  
  
1.Validate Name  
2.Validate Content  
Enter your choice:  
**2**  
Enter the content to be validated:  
**Buy the groceries,necessities.**  
Content is valid  
  
**Sample Input and Output 4:**  
  
1.Validate Name  
2.Validate Content  
Enter your choice:  
**2**  
Enter the content to be validated:  
**Buy Groceries in "The mall".**  
Content is invalid

**Note Book - Requirement 4**

**Requirement 4:**  
In this requirement develop a feature in which you can search a List of Notes by name,createdDate,priorityLevel.  
  
a) Create a Class **Note** with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_name | string |
| \_content | string |
| \_size | double |
| \_priorityLevel | double |
| \_receivedDate | DateTime |

Mark all the attributes as private.  
Include appropritae properties.  
Add a default constructor and a parameterized constructor to take in all attributes in the given order:  
Note(string \_name, string \_content, double \_size, double \_priorityLevel,DateTime \_createdDate)  
  
b) Create a class **NoteBO**with the following methods,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| public List<Note> FindNote(List<Note> noteList,string name) | This method accepts a list of notes and name as arguments and returns a list of notes that matches with given name. |
| public List<Note> FindNote(List<Note> noteList,DateTime createdDate) | This method accepts a list of notes and created date as arguments and returns a list of notes that were created on the given specified date. |
| public List<Note> FindNote(List<Note> noteList,double priorityLevel) | This method accepts a list of notes and priorityLevel as arguments, then find all the notess with the given priorityLevel from the note list and return the list of notes with the specified priorityLevel. |

The note details should be given as a comma-separated value in the below order,  
\_name,\_content, \_size,\_priorityLevel,\_createdDate

Create a class named as **Program**, which contains **Main** method. All the input and output operations are done in this method.  
It is used to access the above classes and its method to do this requirement.  
  
**Problem Overview:**  
The note details is a string, which is comma seperated. Split that string and create a Note object and add that note to the noteList(which is maintain in **Main** method).  
After that display the menu,

1.By Name

2.By Created Date

3.By Priority Level  
  
If the user select option-1, then get the name of the note from the user, and find the notes from the noteList using the \_name name(use FindNote(List<Note>,string) method).  
  
If the user select option-2, then get the date of the note created from the user, and find the notes from the noteList using the \_createdDate name(use FindNote(List<Note>,DateTime) method).  
  
If the user select option-3, then get the priorityLevel for the note from the user, and find the notes from the noteList using the \_priorityLevel name(use FindNote(List<Note>,double) method).  
  
Print the note list using the following format:

Print format:  
**Console.WriteLine("{0,-15} {1,-20} {2,-5} {3,-15} {4}","Name","Content","Size","Priority Level","Created Date");**  
  
**Note:**The note lists are displayed in the main method.  
              If any other choice is selected, display "**Invalid Choice**"  
              If search detail is not found, display "**No such note is present**"  
              Display one digit after the decimal point for Double Datatype.  
  
**Sample Input and Output 1:**

Enter the number of Notes:

**4**

**IPL Schedule,CSKPlayingDates,25.0,5.0,27-01-2018**

**Playlist,List of songs,50.5,4.2,02-02-2018**

**OfficeWorks,List of Meetings,23.7,4.4,30-01-2018**

**DaysToRemember,Birth Dates,12.6,4.0,28-01-2018**

Enter a search type:

1.By Name

2.By Created Date

3.By Priority Level

**1**

Enter the Name:

**OfficeWorks**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Content | Size | Priority Level | Created Date |
| OfficeWorks | List of Meetings | 23.7 | 4.4 | 30-01-2018 |

**Sample Input and Output 2:**

Enter the number of Notes:

**4**

**IPL Schedule,CSKPlayingDates,25.0,5.0,27-01-2018**

**Playlist,List of songs,50.5,4.2,02-02-2018**

**OfficeWorks,List of Meetings,23.7,4.4,30-01-2018**

**DaysToRemember,Birth Dates,12.6,4.0,28-01-2018**

Enter a search type:

1.By Name

2.By Created Date

3.By Priority Level

**2**

Enter the Created Date:

**27-01-2018**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Content | Size | Priority Level | Created Date |
| IPL Schedule | CSKPlayingDates | 25.0 | 5.0 | 27-01-2018 |

**Sample Input and Output 3:**  
Enter the number of Notes:

**4**

**IPL Schedule,CSKPlayingDates,25.0,5.0,27-01-2018**

**Playlist,List of songs,50.5,4.2,02-02-2018**

**OfficeWorks,List of Meetings,23.7,4.4,30-01-2018**

**DaysToRemember,Birth Dates,12.6,4.0,28-01-2018**

Enter a search type:

1.By Name

2.By Created Date

3.By Priority Level

**3**

Enter the Priority Level:

**4.2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Content | Size | Priority Level | Created Date |
| Playlist | List of songs | 50.5 | 4.2 | 02-02-2018 |

**Sample Input and Output 4:**

Enter the number of Notes:

**4**

**IPL Schedule,CSKPlayingDates,25.0,5.0,27-01-2018**

**Playlist,List of songs,50.5,4.2,02-02-2018**

**OfficeWorks,List of Meetings,23.7,4.4,30-01-2018**

**DaysToRemember,Birth Dates,12.6,4.0,28-01-2018**

Enter a search type:

1.By Name

2.By Created Date

3.By Priority Level

**4**

Invalid choice

**Note Book -Requirement 5**

**Requirement 5:**  
  
In this requirement, you need to sort the list of notes based on name, priorityLevel, or createdDate.  
  
a) Create a Class **Note**with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_name | string |
| \_content | string |
| \_size | double |
| \_priorityLevel | double |
| \_createdDate | DateTime |

Mark all the attributes as private.  
Include appropriate properties.  
Add a default constructor and a parameterized constructor to take in all attributes in the given order: Note(string \_name, string \_content, double \_size, double \_priorityLevel,DateTime \_createdDate)  
  
b) Create the following static methods in the **Note** class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| static Note CreateNote(string detail) | This method accepts a String. The note detail separated by commas is passed as the argument. Split the details and create a note object and returns it. |

The note details should be given as a comma-separated value in the below order,  
\_name, \_content, \_size, \_priorityLevel, \_createdDate  
  
c) The **Note**class should implement the **IComparable** interface which sorts the Note list based on \_name attributes.While comparing, all the \_name attributes in the list are unique.  
  
d) Create a class **DateComparer** which implements **IComparer** interface and sort the Note list based on \_createdDate. While comparing, all the \_createdDate attributes in the list are unique.  
  
e) Create a class **PriorityLevelComparer** which implements **IComparer** interface and sort the Note list based on the \_priorityLevel. While comparing, all the  \_priorityLevel  attributes in the list are unique.  
  
Get the number of Note and note details and create a note list. Sort the note according to the given option and display the list.  
  
**Problem Overview:**   
In this requirement, which has the menu driven,   
Enter a type to sort:

1.Sort by name

2.Sort by priority level

3.Sort by date created

The first line of the input consist of an integer, that corresponds to the number of notes n.  
The next n line of the input consists of a string, that corresponds to the note details(which is comma seperated string).  
The format for note is ( \_name, \_content, \_size, \_priorityLevel, \_createdDate ).   
  
If the user select choice-1, then sort(ascending) the note list using  \_name of the note, and also display the note list.  
If the user select choice-2, then sort(ascending) the note list using  \_priorityLevel of the note, and also display the note list.  
If the user select choice-3, then sort(ascending) the note list using  \_createdDate of the note, and also display the note list.  
  
Print format:  
**Console.WriteLine("{0,-15} {1,-20} {2,-5} {3,-15} {4}", "Name","Content","Size","Priority level","Created date");**  
Display one digit after decimal point for double datatype.  
  
**Sample Input and Output 1:**

Enter the number of the notes:

**5**

**Mobiles,Purchasing Choices,10.1,4.9,12-12-2017**

**ImportantDates,Birth Dates,12.6,3.5,28-01-2018**

**OfficeWorks,List of Meetings,23.7,4.0,30-01-2018**

**Schedule,Works to be done,15.3,4.8,12-12-2018**

**Invitation,List of Recipients,25.8,4.6,01-02-2018**

Enter a type to sort:

1.Sort by name

2.Sort by priority level

3.Sort by date created

**1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Content | Size | Priority level | Created date |
| ImportantDates | Birth Dates | 12.6 | 3.5 | 28-01-2018 |
| Invitation | List of Recipients | 25.8 | 4.6 | 01-02-2018 |
| Mobiles | Purchasing Choices | 10.1 | 4.9 | 12-12-2017 |
| OfficeWorks | List of Meetings | 23.7 | 4.0 | 30-01-2018 |
| Schedule | Works to be done | 15.3 | 4.8 | 12-12-2018 |

**Sample Input and Output 2:**

Enter the number of the notes:

**5**

**Mobiles,Purchasing Choices,10.1,4.9,12-12-2017**

**ImportantDates,Birth Dates,12.6,3.5,28-01-2018**

**OfficeWorks,List of Meetings,23.7,4.0,30-01-2018**

**Schedule,Works to be done,15.3,4.8,12-12-2018**

**Invitation,List of Recipients,25.8,4.6,01-02-2018**

Enter a type to sort:

1.Sort by name

2.Sort by priority level

3.Sort by date created

**2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Content | Size | Priority level | Created date |
| ImportantDates | Birth Dates | 12.6 | 3.5 | 28-01-2018 |
| OfficeWorks | List of Meetings | 23.7 | 4.0 | 30-01-2018 |
| Invitation | List of Recipients | 25.8 | 4.6 | 01-02-2018 |
| Schedule | Works to be done | 15.3 | 4.8 | 12-12-2018 |
| Mobiles | Purchasing Choices | 10.1 | 4.9 | 12-12-2017 |

**Sample Input and Output 3:**

Enter the number of the notes:

**5**

**Mobiles,Purchasing Choices,10.1,4.9,12-12-2017**

**ImportantDates,Birth Dates,12.6,3.5,28-01-2018**

**OfficeWorks,List of Meetings,23.7,4.0,30-01-2018**

**Schedule,Works to be done,15.3,4.8,12-12-2018**

**Invitation,List of Recipients,25.8,4.6,01-02-2018**

Enter a type to sort:

1.Sort by name

2.Sort by priority level

3.Sort by date created

**3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Content | Size | Priority level | Created date |
| Mobiles | Purchasing Choices | 10.1 | 4.9 | 12-12-2017 |
| ImportantDates | Birth Dates | 12.6 | 3.5 | 28-01-2018 |
| OfficeWorks | List of Meetings | 23.7 | 4.0 | 30-01-2018 |
| Invitation | List of Recipients | 25.8 | 4.6 | 01-02-2018 |
| Schedule | Works to be done | 15.3 | 4.8 | 12-12-2018 |

**Note Book - Requirement 6**

**Requirement 6:**  
  
In this requirement, given a list of notes you need to find the number of notes created each day using SortedDictionary.  
  
a) Create a Class **Note**with the following attributes:

|  |  |
| --- | --- |
| **Member Field Name** | **Type** |
| \_id | string |
| \_content | string |
| \_size | double |
| \_priorityLevel | double |
| \_createdDate | DateTime |

Mark all the attributes as private.  
Include appropriate properties.  
Add a default constructor and a parameterized constructor to take in all attributes in the given order: Note(string \_name, string \_content, double \_size, double \_priorityLevel,DateTime \_createdDate)  
  
b) Create the following static methods in the **Note**class,

|  |  |
| --- | --- |
| **Method Name** | **Description** |
| static SortedDictionary<DateTime,int> CalculateDateCount(List<Note> list) | This method accepts a list of Note as arguments and returns a SortedDictionary with the createdDate as key and number of notes created on that date as value and returns the SortedDictionary. |

In the SortedDictionary have the \_createdDate as key and Count the number of notes created on the date and keep the number of notes as value. Print the value sorted by Date.  
  
The note details should be given as a comma separated value in the below order,  
\_name,\_content,\_size,\_priorityLevel,\_createdDate  
  
Create a class named as   **Program**  , which contains   **Main**   method, All the input and output operations are occurs in this method.    
It is also used to access the above class and its method to done this requirement.  
  
**Problem Overview:**  
The first line of the input consist of an integer, that corresponds to the number of notes n.  
The next n line of the input consists of a string, that corresponds to the note details(which is comma seperated string).  
The format for note is ( \_name,\_content,\_size,\_priorityLevel,\_createdDate ).   
  
Print format:  
**Console.WriteLine("{0,-15} {1}","Date","Count");**  
  
**Sample Input and Output 1:**

Enter the number of notes:

**4**

**Mobiles,Purchasing Choices,10.1,4.9,12-12-2018**

**Grocery,Items to be bought,20.5,4.5,03-02-2018**

**Schedule,Works to be done,15.3,4.0,03-02-2018**

**Invitation,List of Recipients,25.8,4.6,01-02-2018**

|  |  |
| --- | --- |
| Date | Count |
| 01-02-2018 | 1 |
| 03-02-2018 | 2 |
| 12-12-2018 | 1 |