Exercises: Implement Hash Table with Chaining

This document defines the **in-class exercises** assignments for the "Data Structures" course @ Software University.

Part I. Events in Given Date Range

Write a program that reads a set of events in format "**Event name | Date and time**" and a series of date ranges **a < b** and prints for each range (a < b) all events within the range [a … b] inclusively (ordered by date; for duplicated dates preserve the order of appearance).

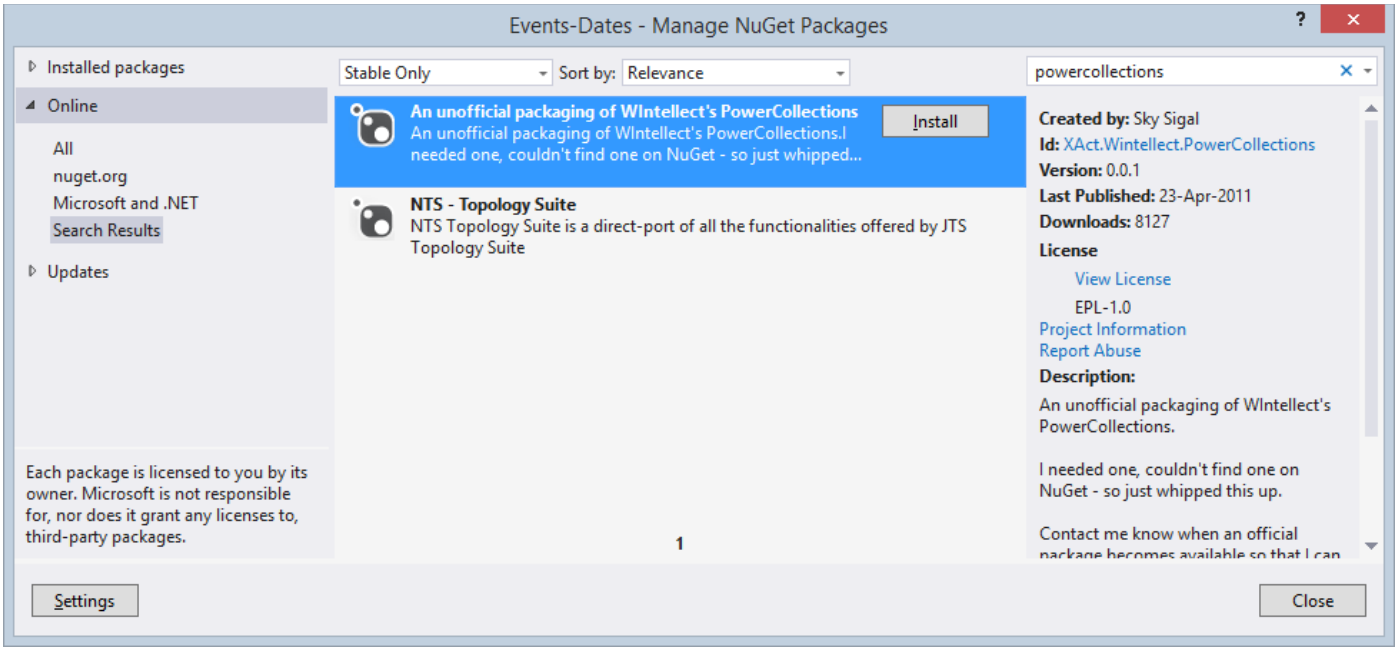
Examples

Input	Output
5 C# Course – Group II | 15-Aug-2015 14:00 Data Structures Course | 13-Aug-2015 18:00 C# Course – Group I | 15-Aug-2015 10:00 Seminar for Java Developers | 18-Aug-2015 19:00 Game Development Seminar | 15-Aug-2015 10:00 2 15-Aug-2015 10:00 | 1-Sep-2015 0:00 13-Aug-2015 10:00 | 13-Aug-2015 20:00	4 C# Course - Group I | 15-Aug-2015 Game Development Seminar | 15-Aug-2015 C# Course - Group II | 15-Aug-2015 Seminar for Java Developers | 18-Aug-2015 1 Data Structures Course | 13-Aug-2015 18:00

Problem 1. Reference the Power Collections from NuGet

Start Visual Studio (or your favorite C# development IDE). Create a new C# project (Console application).

Reference Wintellect Power Collections from NuGet:



Problem 2. Read All Events in Ordered Multi-Dictionary

Read all events in an ordered multi-dictionary:

```
Thread.CurrentThread.CurrentCulture = CultureInfo.InvariantCulture;

var events = new OrderedMultiDictionary<DateTime, string>(true);
int n = int.Parse(Console.ReadLine());
for (int i = 0; i < n; i++)
{
    string eventEntry = Console.ReadLine();
    var eventTokens = eventEntry.Split('|');
    string eventName = eventTokens[0].Trim();
    DateTime eventDate = DateTime.Parse(eventTokens[1].Trim());
    events.Add(eventDate, eventName);
}
```

Initially, we **reset the current culture** (locate) to ensure the system locale in the operating system regional settings does not affect the date and time format (we want to use the neutral date and time format).

Then, we **create an ordered multi-dictionary**: **`OrderedMultiDictionary<DateTime, string>`**. It maps the event dates and times to event names.

Finally, we read the input line by line and put the events from each line into the multi-dictionary.

Problem 3. Find the Events in the Given Dates Range

Write some code to efficiently take a subrange from the ordered multi-dictionary. Read the start and end dates from the console and then use **`.Range(startDate, true, endDate, true)`** method:

```
DateTime startDate = DateTime.Parse(Console.ReadLine());
DateTime endDate = DateTime.Parse(Console.ReadLine());
var eventsInRange = events.Range(startDate, true, endDate, true);
```

Problem 4. Print the Results

Finally, print the expected output:

```
Console.WriteLine(eventsInRange.KeyValuePairs.Count);
foreach (var e in eventsInRange)
{
    foreach (var eventName in e.Value)
    {
        Console.WriteLine("{0} | {1:dd.MM.yyyy}", eventName, e.Key);
    }
}
```

Problem 5. Put It All Togethder

Write the logic to process a series of date ranges, not just a single, to produce the requested output.

Part II. Rope for Efficient String Editing

You have to implement a string editor that starts from empty string and executes sequence of commands:

- **`INSERT some_string`** – inserts given string at front of the text. Print "**OK**" as command result.
- **`APPEND some_string`** – appends given string at the end of the text. Print "**OK**" as command result.
- **`DELETE start_index count`** – deletes the specified substring. Print "**OK**" as command result in case of success. Print "**ERROR**" in case of invalid substring.
- **`PRINT`** – prints the string in the editor.

Read the commands from the console, process them and finally print the results. Do not print the results until all commands are processed.

Ensure your programs runs **efficiently** for tens of thousands of commands.

Hint: use **rope of chars**.