1.4 Abhängigkeiten auf die Reihe bekommen

Entwickle eine Klasse, die einen Plan zur Abarbeitung untereinander abhängiger Jobs erstellt.
Jeder Job wird durch einen Buchstaben repräsentiert, z.B. ”c” oder ”x”. Es kann sein, dass vor einem Job ”a” ein Job ”u” abgearbeitet werden muss. Dann ist ”a” von ”u” abhängig. Natürlich kann jeder Job von beliebig vielen anderen abhängig sein und die wiederum von anderen...
Nachdem eine Reihe von Jobs mit ihren Abhängigkeiten registriert wurden, soll die Klasse sie in eine Abarbeitungsreihenfolge bringen, in der Abhängige nach denen dran sind, von denen sie abhängen.

Die Schnittstelle sieht wie folgt aus:

```
class OrderedJobs {
public:
  void Register(char dependentJob, char independentJob);
  void Register(char job);
  string Sort();
}
```

Wenn dann wie folgt registriert wird...

```
OrderedJobs jobs;
jobs.Register('c');
jobs.Register('b', 'a');
jobs.Register('c', 'b');
```

...dann liefert `jobs.Sort()` dieses Ergebnis:

```
"abc"
```

Jobs, die in mehreren Registrierungen vorkommen, tauchen in der Sortierung nur einmal auf.
Unabhängige Jobs können in beliebiger Reihenfolge in der Sortierung stehen, solange sie vor denen abgearbeitet werden, die von ihnen abhängig sind.
Direkte oder indirekte zirkuläre Abhängigkeiten sollen über eine Ausnahme(Exception) gemeldet werden – spätestens bei der Sortierung.