1. (a) Bestimmen Sie die Äquivalenzklassen zur Berechnung folgender Rabatt-Funktion. Beachten Sie dabei auch Grenzwerte, und Sonderfälle. Geben Sie tabellarisch die Eingaben der drei Parameter, das Ergebnis und eine kurze Beschreibung der Äquivalenzklasse an. Gliedern Sie Ihre Äquivalenzklassen wie folgt:

   - Regulären Situationen
   - Fehlerfälle
   - Grenzfälle
   - Sonderfälle

   Extremwerte können Sie hier außer Acht lassen.

   Grenzfälle können sich auf reguläre Situationen und Fehlerfälle beziehen. Hier darf es Überschneidungen (oder Dopplungen) geben. Bitte geben Sie diese auch unbedingt in den verschiedenen Gruppen an.

```java
/**
* Berechnet den Rabatt auf den Eintrittspreis eines Diskobesuchs
* Frauen erhalten 5% Rabatt. Sind diese zusaetzlich unter 30, werden
* zusaetzliche 2% Rabatt gewaehrt.
* Beamte erhalten nochmals 5% Rabatt (unabh. vom Geschlecht).
*
* @param geschlecht   Wert eines enums mit den Auspraegungen "maennlich"
*   und "weiblich"
* @param alter        Das Alter (17 < alter < 80) des Diskobesuchers,
*   ansonsten wird eine Exception geworfen
* @param istBeamter   Flag, um anzuzeigen, dass der Diskobesucher ein
*   Beamter ist
* @return             Rabattfaktor (1.0 entsprechen 100% = freier Eintritt,
*   0.5 entsprechen 50% = halber Eintrittspreis)
* @throws             Exception in allen weiteren nicht hier spezifizierten
*   Faellen
*/
public double berechneRabatt(GESCHLECHT geschlecht,
                                          int alter, boolean istBeamter){
```

Lösung:

Betrachtung der Äquivalenzklassen:

Nr.	Äquivalenzklasse	Typ	Repräsentant		
			`geschlecht`	`alter`	`istBeamter`
0	Rabatt 0%	(Zulässig, 0.0)	`maennlich`	`42`	`false`
1	Rabatt 5%	(Zulässig, 0.05)	`maennlich`	`42`	`true`
2	Rabatt 5%	(Zulässig, 0.05)	`weiblich`	`42`	`false`
3	Rabatt 7%	(Zulässig, 0.07)	`weiblich`	`24`	`false`
4	Rabatt 10%	(Zulässig, 0.1)	`weiblich`	`42`	`true`
5	Rabatt 12%	(Zulässig, 0.12)	`weiblich`	`24`	`true`
6	`alter < 18`	(Unzulässig, `Exception`)	egal	`-42`	egal
7	`alter > 79`	(Unzulässig, `Exception`)	egal	`1337`	egal
8	`geschlecht` $\lightning$	(Unzulässig, `Exception`)	`elmo`	egal	egal

Grenzwertbetrachtung:

Nr.	Typ	Repräsentant		
		`geschlecht`	`alter`	`istBeamter`
6, (8)	Unzulässig	egal	`17`	egal
0, 1, 2, 3, 5, (8)	Zulässig	egal	`18`	egal
0, 2, 4, 5, (8)	Zulässig	egal	`79`	egal
7, (8)	Unzulässig	egal	`80`	egal

Sonderfallbetrachtung:

Ergibt hier erstaunlich wenig Sinn, da es für die Parameter `geschlecht` und `istBeamter` ausschließlich Sonderfälle gibt, da beide lediglich zweielementig sind und die möglichen Werte von `alter` bereits in der Grenzwertbetrachtung zur Genüge abgedeckt und insbesondere genügend gleichverteilt sind.

(b) Erklären Sie allgemein was Extremwerte sind und geben Sie ein Beispiel dazu.

Lösung:

Markus Lausberg und Robin Ziehe fassten es unter Aufsicht von Prof. Kraft gut zusammen:

> Extremwerte sind Werte an den Rändern von Definitionsbereichen. Sie werden oft gesondertbehandelt und sollten deshalb auf jeden Fall getestet werden, da Programmabstürze oft mit diesen Werten zusammenhängen und dies dringend vermieden werden muss.

Man betrachte die folgende sehr komplexe Funktion:

```
/*
* Berechnet den Nachfolger der gegeben Zahl

* @param num    Zahl die erhoeht werden soll
* @return       Nachfolger der Zahl
*/
public byte succ(byte num) {
    // ...
}
```

Offensichtlich erhält man bei dem Aufruf `succ(127)` einen Schrecken, wenn man entsetzt feststellen muss, dass auf einmal `0` der Nachfolger zu `127` ist.

Natürlich sind wir aber klüger und haben `127` als Extremwert identifiziert und entsprechend behandelt und können damit beruhigt schlafen gehen, ohne dass uns das Quality Assurance-Team feuern lässt.