3 Scheduling (25)

a) Rate-Monotonic Scheduling (12)

Task	Laufzeit (ms)	Periode (ms)
T1	2	20
T2	2	10
T3	3	13
T4	2	12
T5	3	23

Folgendes wird angenommen:

- Alle Tasks sind periodisch
- Deadline = Periode
- Scheduling Overhead wird vernachlässigt
- Alle Tasks sind unabhängig

Ist dieses Taskset nach der notwendigen Scheduling-Bedingung unter Verwendung von Rate-Monotonic Scheduling (RMS) schedulbar?

◯ Ja ◯ Nein

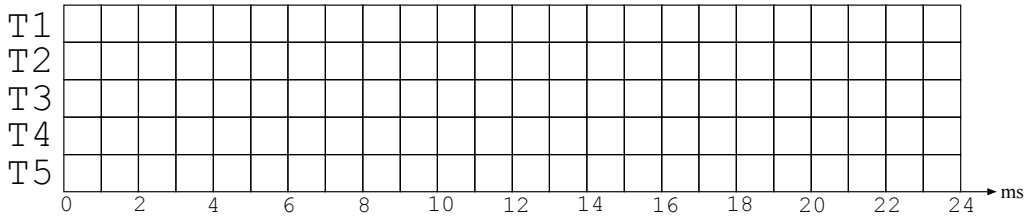
Bitte begründen Sie Ihre Antwort:

Ist dieses Taskset nach der hinreichenden Scheduling-Bedingung unter Verwendung von RMS schedulbar?

◯ Ja ◯ Nein

Bitte begründen Sie Ihre Antwort:

Das oben angegebene Taskset soll mit einem Rate-Montonic Scheduling Algorithmus für den Worst Case[1] gescheduled werden. Bitte vervollständigen Sie den gesamten Zeitbereich der folgenden Skizze bzw. kennzeichnen Sie eine Verletzung einer Deadline.



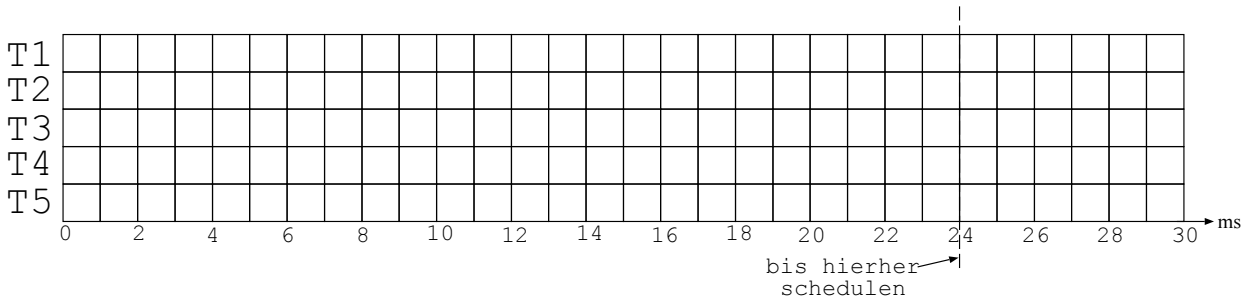
[1] Im Worst Case beginnen alle Perioden bei 0.

b) Earliest Deadline First Scheduling (EDF) (8)

Schedulen Sie das gegebene Task Set nach dem Earliest Deadline First Verfahren *mit Preemption* bis zu $t = 24$ ms. Als Arrival Time können Sie für alle Tasks den Zeitpunkt 0 annehmen, der Scheduling Overhead ist wiederum 0. Weiters sind die angegebenen Deadlines zu beachten. Sollte das Taskset nicht gescheduled werden können, so zeichnen Sie bitte die Verletzung der Deadline eindeutig ein.

Die Deadlines für die Tasks sind:

Task	Laufzeit (ms)	Periode (ms)	Deadline (ms)
T1	2	20	12
T2	2	10	10
T3	3	13	13
T4	2	12	7
T5	3	23	8



c) Fragen zu Scheduling (5)

Wenn ein Test unter Verwendung von Rate-Monotic Scheduling aufgrund der notwendingen Scheduling-Bedingung ergibt, dass ein Taskset schedulbar ist, aber ein Test aufgrund der hinreichenden Scheduling-Bedingung ergibt, dass ein Taskset *nicht* schedulbar ist, könnte es trotzdem sein, dass das Taskset schedulbar ist oder ist das unmöglich?

◯ Ja, ein solches Taskset könnte schedulbar sein.

◯ Nein, es ist unmöglich, dass ein solches Taskset schedulbar ist

Kreuzen Sie bei der folgenden Aussage bitte an, ob diese richtig oder falsch ist.

Bei Scheduling nach dem Round-Robin-Verfahren werden I/O-intensive Prozesse gegenüber rechenzeit-intensiven (mit wenig I/O) benachteiligt.

◯ richtig ◯ falsch

Ersatzvorlagen

Bitte streichen Sie die anderen Vorlagen durch, wenn Sie diese Vorlagen verwenden.

2 Scheduling (20)

a) Earliest Deadline First Scheduling (EDF) (10)

Schedulen Sie das gegebene Task Set nach dem Earliest Deadline First Verfahren *mit Preemption* bis zum Zeitpunkt $t = 26$ ms unter Beachtung der angegebenen Deadlines.

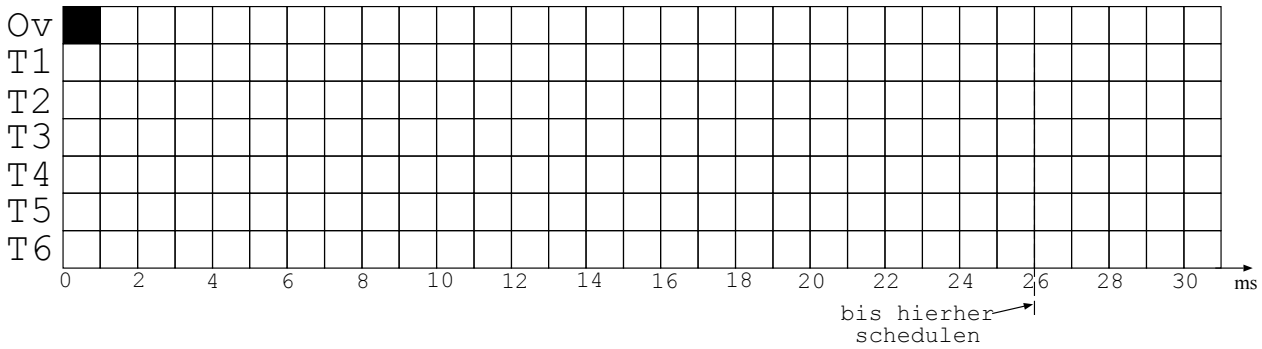
Sollte das Taskset nicht gescheduled werden können, so zeichnen Sie das gültige Schedule bitte bis zum Zeitpunkt der Verletzung der Deadline und kennzeichnen Sie diesen Zeitpunkt.

Das Taskset:

Task	Laufzeit (ms)	Periode (ms)	Deadline (ms)
T1	1	11	9
T2	2	13	10
T3	1	16	16
T4	1	12	7
T5	2	25	14
T6	3	13	12

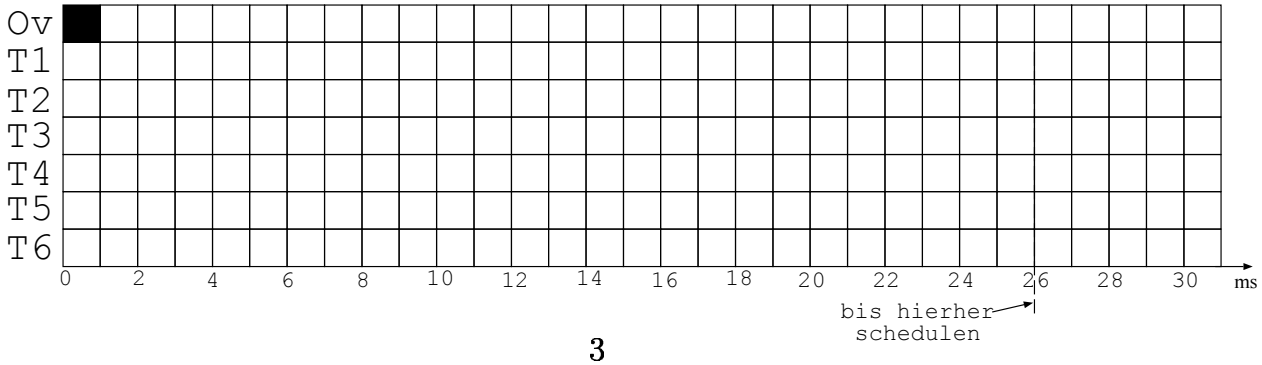
- Alle Tasks sind periodisch.
- Der Scheduling Overhead beträgt 1 ms.
- Alle Tasks sind unabhängig.
- Die erste Ankunftszeit (arrival time) ist für alle Tasks der Zeitpunkt 0.

Bitte schedulen Sie das obige Taskset.



Ersatzvorlage

Bitte streichen Sie die andere Vorlage durch, wenn Sie diese Vorlage verwenden.



b) Fragen zu Scheduling (10)

Werden bei Rate-Monotic Scheduling die Prioritäten den Tasks statisch oder dynamisch zugeordnet?

◯ dynamisch ◯ statisch

Begründung:

Was sind Vorteile von Scheduling-Mechanismen, die die Priorität statisch zuweisen?

Ist Scheduling nach dem First-In-First-Out Prinzip preemptive?

◯ ja ◯ nein

Begründung:

Fragen zu Scheduling nach dem Round-Roubin-Verfahren:

Was sind die Vorteile von großen Zeitquanten?

Was sind die Nachteile von großen Zeitquanten?

Was sind die Vorteile von kleinen Zeitquanten?

Was sind die Nachteile von kleinen Zeitquanten?

4 Scheduling (20)

Real-Time Scheduling

Klassifizieren Sie die Scheduling-Strategien Rate Monotonic Scheduling (RMS) und Earliest Deadline First (EDF) anhand der folgenden Eigenschaften:

- "preemptive" versus "non-preemptive" Scheduling-Verfahren
- zentrale (centralized) versus verteilte (distributed) Scheduling-Entscheidung (Entscheidung der Abarbeitung von Prozessen)
- dynamisches (= Task Set zur Laufzeit veränderbar) versus statisches (= fixiertes Task Set; Abarbeitung der Tasks wird off-line entschieden) Scheduling
- dynamische versus statische Zuweisung von Prioritäten

Bitte klassifizieren Sie EDF und RMS durch Eintragen von Kreuzen (Kreuze haben die Bedeutung "diese Eigenschaft trifft zu"):

Eigenschaft	**RMS**	**EDF**
preemptive		
nonpremtive		
zentrale Entscheidung		
verteilte Entscheidung		
dynamisches Scheduling		
statisches Scheduling		
dynamische Prioritätenzuweisung		
statische Prioritätenzuweisung		

Kreuzen Sie an, ob die folgenden Aussagen korrekt oder inkorrekt sind:

Scheduling nach dem Round-Roubin Prinzip wird bei Verwendung sehr großer Zeitquanten zu Scheduling nach dem First-In-First-Out Prinzip.

◯ korrekt ◯ inkorrekt

Eine Scheduling-Strategie für eine CPU ist "preemptive", wenn das Betriebssystem einen Prozess *nicht* von der CPU suspendieren kann.

◯ korrekt ◯ inkorrekt

Echtzeitsysteme benutzen generell preemptive CPU Scheduling Verfahren.

◯ korrekt ◯ inkorrekt

Die Antwortenzeiten von Systemen, die "preemptive" Scheduling-Strategien benutzen, sind besser voraussagbar (=Angaben sind zuverlässiger), als die Antwortzeiten von Systemen, die kooperatives (= "non-preemptive") Scheduling benutzen.

◯ korrekt ◯ inkorrekt

Generell kann man sagen, dass rechenintensive Prozesse bei Scheduling nach dem Round-Robin-Prinzip durchschnittlich schneller abgearbeitet werden als I/O-intensive Prozesse.

◯ korrekt ◯ inkorrekt

Scheduling-Strategien, die Prioritäten statisch zuweisen, sind leichter zu implementieren als Scheduling-Strategien, die Prioritäten dynamisch zuweisen.

◯ korrekt ◯ inkorrekt

Scheduling-Strategien, die Prioritäten statisch zuweisen, verlangen mehr Laufzeit-Overhead als Scheduling-Strategien, die Prioritäten dynamisch zuweisen.

◯ korrekt ◯ inkorrekt

Scheduling nach dem Multi-level Feedback Queue Prinzip bevorzugt Prozesse mit langen Ausführungszeiten.

◯ korrekt ◯ inkorrekt

Scheduling nach dem Multi-level Feedback Queue Prinzip bevorzugt I/O-intensive Prozesse, um gute I/O-Geräte-Auslastung zu erreichen.

◯ korrekt ◯ inkorrekt

Deadline Scheduling verlangt einen intensiven Ressourcen-Einsatz; deshalb ist der Overhead im Vergleich zu anderen Scheduling Strategien groß.

◯ korrekt ◯ inkorrekt

Scheduling nach dem Round-Robin-Prinzip unter Verwendung von langen Zeitquanten bevorzugt I/O-intensive Prozesse gegenüber rechenzeitintensiven Prozessen.

◯ korrekt ◯ inkorrekt

Eine Deadline-Verletzung eines "Soft Real-Time" Prozesses hat fatale Folgen; deshalb muss eine solche Deadline von einem Scheduling-Algorithmus auf jeden Fall eingehalten werden.

◯ korrekt ◯ inkorrekt

2 Scheduling (20)

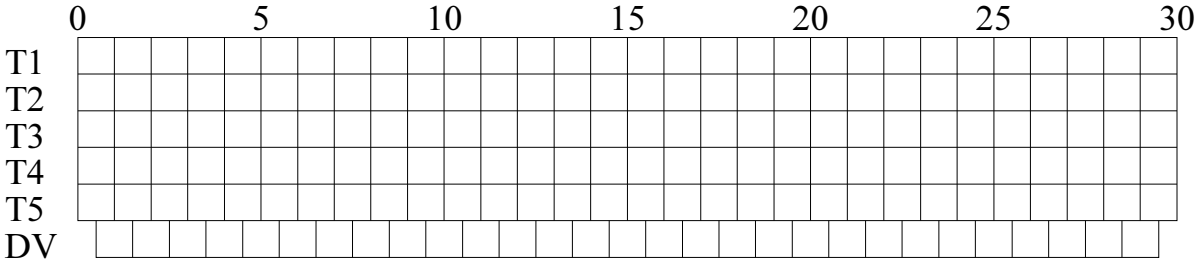
a) Rate-Monotonic Scheduling (10)

Task	Laufzeit (ms)	Periode (ms)
T1	5	12
T2	2	23
T3	2	22
T4	2	11
T5	3	13

Folgendes wird angenommen:

- Alle Tasks sind periodisch.
- Deadline = Periode.
- Laufzeit ist bekannt.
- Der Scheduling Overhead wird vernachlässigt.
- Alle Tasks sind unabhängig.

Dieses Taskset soll mit einem Rate-Montonic Scheduling Algorithmus für den Worst Case[1] gescheduled werden. Bitte vervollständigen Sie die folgende Skizze zur Gänze bzw. bis zur ersten Deadlineverletzung. Kennzeichnen Sie eine solche, indem Sie in der Zeile DV den Task eintragen, für den die Deadline nicht eingehalten wurde.



Bitte beantworten Sie diese Frage unabhängig von Ihrer Lösung oberhalb: Nehmen Sie an, dass das obige Taskset gescheduled werden kann. Bis zu welchem Zeitpunkt müssen Sie das angegebene Taskset *mindestens* schedulen, um sicher gehen zu können, dass es wirklich schedulbar ist. Bitte richtige Antworten ankreuzen:

◯ 11 ms ◯ 12 ms ◯ 13 ms ◯ 22 ms ◯ 23 ms
◯ 24 ms ◯ 26 ms ◯ 44 ms ◯ 46 ms
◯ einen Zeitpunkt der in der Liste nicht aufscheint

[1] Im Worst Case beginnen alle Perioden bei 0.

b) Fragen zu Rate-Monotonic Scheduling (6)

Gegeben seien $n = 4$ Tasks; mit den Ausführungszeiten C_1, C_2, C_3, C_4 und den Perioden T_1, T_2, T_3, T_4; der Scheduling-Overhead wird vernachlässigt:

Welche Bedingung muss erfüllt sein, dass Sie sicher **kein** mögliches Scheduling finden können?

Welche Bedingung müssen die 4 Tasks erfüllen, dass es sicherlich ein mögliches Scheduling gibt?

Können Sie mit Hilfe dieser beiden Bedingungen alle Fälle klassifizieren, oder gibt es Fälle welche sich nicht eindeutig entscheiden lassen?

◯ alle Fälle sind eindeutig entscheidbar
◯ es gibt Fälle, die nicht eindeutig entscheidbar sind

In einem System das RMS verwendet, messen Sie eine Prozessorauslastung von 0,65. Die Taskanzahl ist Ihnen unbekannt; weiters wird der Scheduling-Overhead vernachlässigt. Können Sie sicher sein, dass alle Deadlines eingehalten werden?

◯ ja ◯ nein

Wenn Sie die vorherige Frage mit nein beantwortet haben, geben Sie bitte an, welche Informationen Sie zusätzlich benötigen, um eine sichere Aussage machen zu können. Im Fall, dass Sie mit ja geantwortet haben, erklären Sie bitte, warum diese Information ausreicht?

c) Fragen zu Scheduling allgemein (4)

Kreuzen Sie bitte an ob die Aussagen korrekt sind

Bei First-Come-First-Served (FCFS) Scheduling kann es niemals zu Starvation kommen:

◯ korrekt ◯ inkorrekt

Round Robin (RR) Scheduling erfordert nur geringen Scheduling-Overhead:

◯ korrekt ◯ inkorrekt

Shortest-Remaining-Time (SRT) Scheduling benachteiligt Prozesse mit langen Ausführungszeiten:

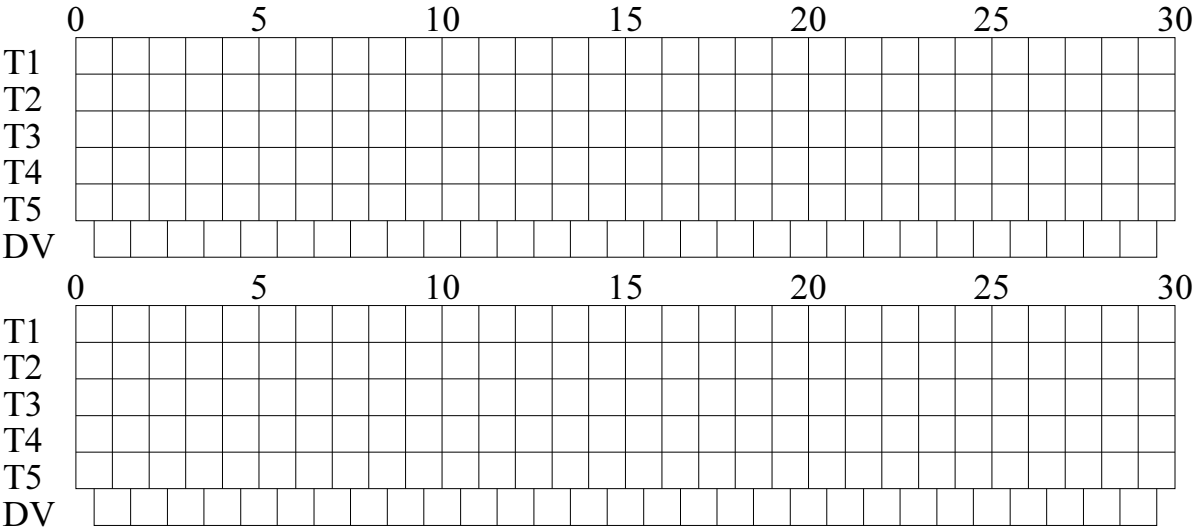
◯ korrekt ◯ inkorrekt

Bei Highest-Response-Ratio-Next (HRRN) Scheduling ist Starvation unmöglich:

◯ korrekt ◯ inkorrekt

Ersatzvorlagen zu a)

Streichen Sie ungültige Vorlagen deutlich durch, wenn Sie eine dieser Vorlagen verwenden!



4 Scheduling (22)

4.1 Single Processor Scheduling (13)

Gegeben ist nebenstehendes Taskset. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar.

Task	Ausführungszeit	Periodendauer
A	2	8
B	1	4
C	2	7
D	1	5

Ermitteln Sie für dieses Taskset die *notwendige* und die *hinreichende* Bedingung für das *Rate Monotonic Scheduling* (RMS) Verfahren. Berechnen Sie die Zahlenwerte auf mindestens eine Kommastelle genau.

Ist die notwendige Bedingung erfüllt? 1P ◯ Ja ◯ Nein
Ist die hinreichende Bedingung erfüllt? 1P ◯ Ja ◯ Nein

Versuchen Sie das Taskset einmal mit dem RMS und einmal mit dem *Earliest Deadline First* (EDF) Verfahren zu schedulen. Verwenden Sie dazu die nachstehenden Vorlagen. Tragen Sie bei jeder Vorlage die aktiven Taskzeiten ein und bezeichnen Sie deutlich eventuelle Deadlineverletzungen. Eine Vorlage dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

Scheduling nach dem **RMS**-Verfahren: ersten 3 spalten: 1 Punkt, spalte 4,5:1P; spalte 6,7: 1P; spalte 8,9:1P

A																		
B																		
C																		
D																		
	0					5						10					15	

Scheduling nach dem **EDF**-Verfahren: 5x (3 spalten: 1Punkt)

A																		
B																		
C																		
D																		
	0					5						10					15	

Ersatzvorlage: Scheduling nach dem -Verfahren:

A																		
B																		
C																		
D																		
	0					5						10					15	

4.2 Verständnisfragen (9)

Pro falsche oder fehlende Antwort -1P. Welche Scheduling-Strategie ähnelt einer Round Robin-Strategie mit sehr langen Zeitscheiben?

Beurteilen Sie die folgenden Aussagen! Falsche Antworten werden negativ gewertet!

◯ Ja ◯ Nein Earliest Deadline First Scheduling ist optimal in Single-Prozessor Systemen.

◯ Ja ◯ Nein Earliest Deadline First Scheduling ist optimal in Multi-Prozessor Systemen.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Single-Prozessor Systemen immer eine Lösung.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Single-Prozessor Systemen immer eine Lösung, wenn eine solche existiert.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Multi-Prozessor Systemen immer eine Lösung.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Multi-Prozessor Systemen immer eine Lösung, wenn eine solche existiert.

◯ Ja ◯ Nein Bei Round Robin Scheduling kann das Problem der Starvation nicht auftreten.

◯ Ja ◯ Nein Beim Scheduling nach dem Round-Robin-Verfahren werden I/O-intensive Prozesse benachteiligt.

◯ Ja ◯ Nein Wenn in einem Single-Prozessor-System bei allen Tasks die Deadline gleich ihrer Periode ist, dann liefert RMS Scheduling dasselbe Ergebnis wie EDF Scheduling.

◯ Ja ◯ Nein Die Prioritäten beim RMS Scheduling ergeben sich durch die *Processor Utilization* der Tasks.

◯ Ja ◯ Nein Eine für das RMS Scheduling hinreichende Bedingung stellt auch eine hinreichende Bedingung für das EDF Scheduling dar.

2 Scheduling (25)

Round-Robin und Feedback Scheduling

Schedulen Sie das nebenstehende Taskset. Beachten Sie dabei folgendes: Zu den angegebenen arrival times wird ein Task in die Ready Queue gestellt. Der Task kann frühestens beim nächsten diskreten Zeitpunkt dem Prozessor zugeteilt werden (Beispiel siehe Task A: arrival time$= -1$; Zuteilung $= 0$). Ein eintreffender Task wird immer ans Ende der Ready Queue gestellt.

Task	Arrival Time	Service Time
A	-1	5
B	2	3
C	3	6
D	5	4
E	6	2

Tragen Sie in der Zeile **P** jenen Task ein der für die entsprechende Zeiteinheit dem Prozessor zugeteilt wird. Der restliche Raster stellt die Ready Queue dar. Tragen Sie hier jene Tasks ein die in der Ready Queue stehen (beginnend in der obersten Zeile mit dem Task der als nächstes den Prozessor zugeteilt bekommt, usw., für Feedback-scheduling reihen Sie die höher prioren Queues zuerst). Schedulen Sie das Task Set nach der Round-Robin Methode und nach der Feedback Methode (time-slice $= 1$, die Anzahl der Queues für die Feedback Methode ist nicht beschränkt).

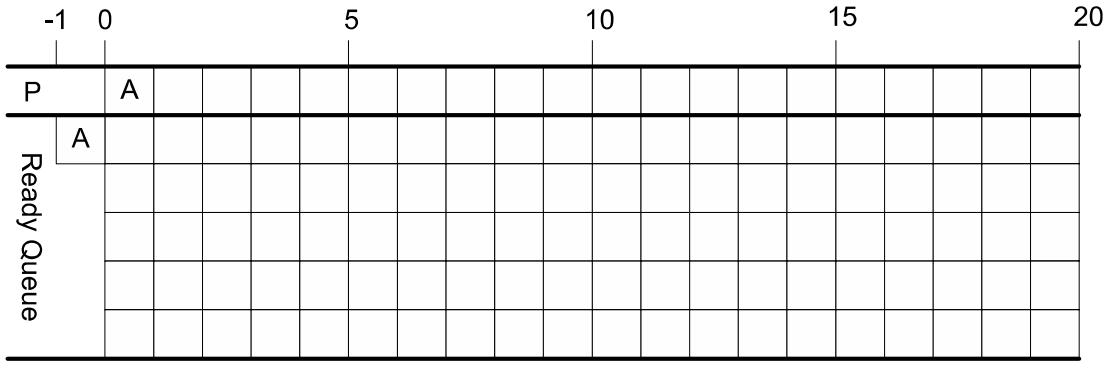


Abbildung 1: Round Robin Scheduling

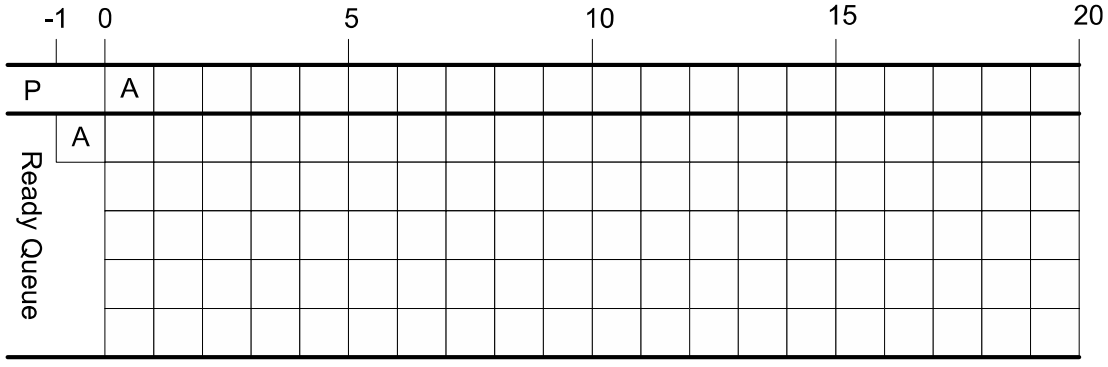


Abbildung 2: Feedback Scheduling

Rate-Monotonic Scheduling

Schedulen Sie das nebenstehende Taskset nach dem Rate-Monotonic Verfahren. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar.

Task	Ausführungszeit	Periodendauer
A	2	6
B	2	12
C	1	4
D	1	5

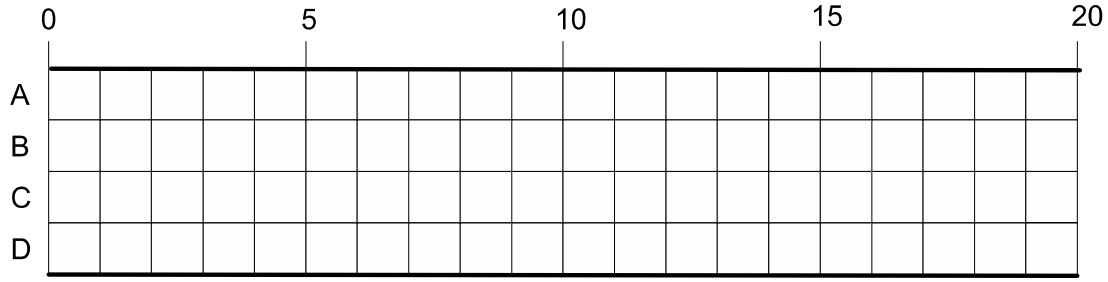


Abbildung 3: Rate-Monotonic Scheduling

Kann es bei diesem Taskset zu einem Deadline-Miss kommen (ja/nein/undefiniert), begründen Sie Ihre Antwort ?

Verständnisfragen

Was bedeutet Starvation?

Geben Sie 3 Scheduling Methoden an bei denen es zu Starvation kommen kann:

Erklären Sie das **gemeinsame** Problem dieser 3 Methoden, wieso es zu Starvation kommen kann:

2 Scheduling (25)

2.1 Uniprocessor Scheduling (13)

Gegeben ist nebenstehendes Taskset. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar.

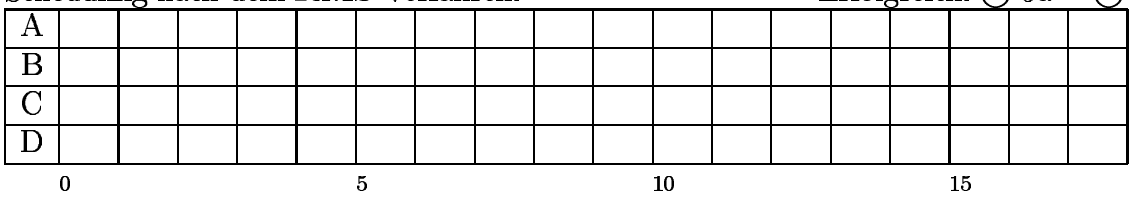
Task	Ausführungszeit	Periodendauer
A	1	5
B	2	9
C	2	7
D	1	4

Ermitteln Sie für dieses Taskset die *notwendige* und die *hinreichende* Bedingung für das *Rate Monotonic Scheduling* (RMS) Verfahren. Berechnen Sie die **Zahlenwerte** überschlagsmäßig (ohne Taschenrechner)!

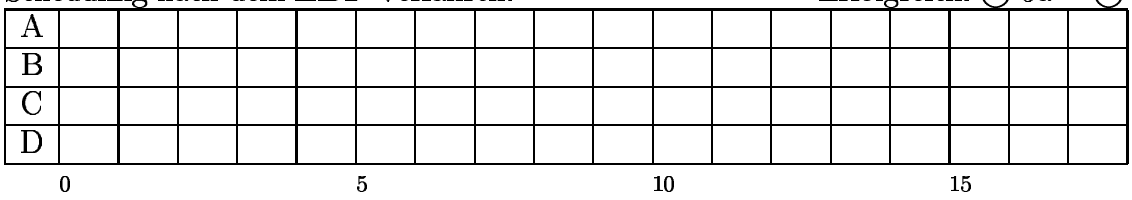
Ist die notwendige Bedingung erfüllt? ◯ Ja ◯ Nein
Ist die hinreichende Bedingung erfüllt? ◯ Ja ◯ Nein

Versuchen Sie das Taskset einmal mit dem RMS und einmal mit dem *Earliest-Deadline-First* (EDF) Verfahren zu schedulen. Verwenden Sie dazu die nachstehenden Vorlagen. Tragen Sie bei jeder Vorlage die aktiven Taskzeiten ein und bezeichnen Sie deutlich eventuelle Deadlineverletzungen. Kreuzen Sie jeweils an ob das Scheduling erfolgreich war. Eine Vorlage dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

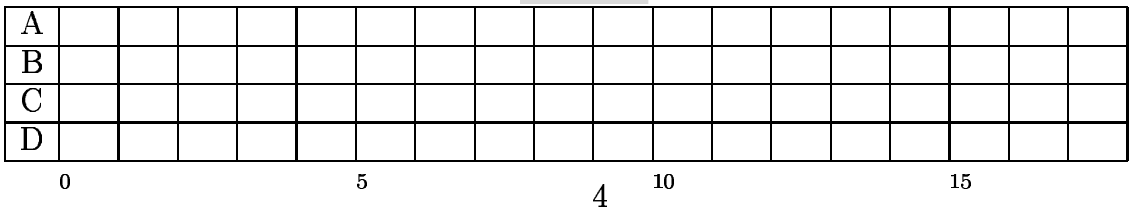
Scheduling nach dem **RMS**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Scheduling nach dem **EDF-**Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Ersatzvorlage: Scheduling nach dem -Verfahren: Erfolgreich: ◯ Ja ◯ Nein



2.2 Verständnisfragen (12)

Beurteilen Sie die folgenden Aussagen! Fehlende Antworten werden negativ, falsche Antworten werden doppelt negativ gewertet!

◯ Ja ◯ Nein Die Prioritäten beim EDF Scheduling ergeben sich durch die Periodendauer der Tasks.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Single-Prozessor Systemen immer eine Lösung, wenn eine solche existiert.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Multi-Prozessor Systemen immer eine Lösung, wenn eine solche existiert.

◯ Ja ◯ Nein Bei Round Robin Scheduling kann das Problem der Starvation auftreten.

◯ Ja ◯ Nein Beim Scheduling nach dem Round-Robin-Verfahren werden I/O-intensive Prozesse benachteiligt.

◯ Ja ◯ Nein Wenn in einem Single-Prozessor-System bei allen Tasks die Deadline gleich ihrer Periode ist, dann liefert RMS Scheduling dasselbe Ergebnis wie EDF Scheduling.

◯ Ja ◯ Nein Eine für das EDF Scheduling hinreichende Bedingung stellt auch eine hinreichende Bedingung für das RMS Scheduling dar.

◯ Ja ◯ Nein Die First-Come-First-Serve-Strategie begünstigt Prozesse mit kurzer Ausführungszeit.

◯ Ja ◯ Nein Das Round-Robin-Verfahren ist preemptive.

◯ Ja ◯ Nein Beim Highest-Response-Ratio-Next-Verfahren kann das Problem der Starvation auftreten.

◯ Ja ◯ Nein Das Shortest-Remaining-Time-Verfahren ist immer preemptive.

◯ Ja ◯ Nein Das Shortest-Process-Next-Verfahren ist immer preemptive.

◯ Ja ◯ Nein Die Prioritäten beim RMS Scheduling ergeben sich durch die reziproke Periodendauer der Tasks.

◯ Ja ◯ Nein Ein Scheduler nach dem Feedback Verfahren benötigt mehr Information pro Task als ein Scheduler nach dem Shortest-Process-Next-Verfahren.

◯ Ja ◯ Nein Earliest Deadline First Scheduling ist optimal in Multi-Prozessor Systemen.

◯ Ja ◯ Nein Das Round-Robin-Verfahren ist optimal, wenn die Periodendauer aller Tasks gleich ist.

◯ Ja ◯ Nein Ein Scheduler nach dem RMS Verfahren benötigt mehr Information pro Task als ein Scheduler nach dem Round-Robin-Verfahren.

VO 182.022 26. Jänner 2005

Prüfung Betriebssysteme

KNr. MNr. Zuname, Vorname

Ges.)(100) 1.)(25) 2.)(25) 3.)(25) 4.)(25) Zusatzblätter:

Bitte verwenden Sie nur dokumentenechtes Schreibmaterial!

1 Scheduling (25)

Round-Robin und Feedback Scheduling

Task	Arrival Time	Service Time
A	0	6
B	1	2
C	3	3
D	4	6
E	8	1
F	16	1

Schedulen Sie das nebenstehende Taskset. Beachten Sie dabei folgendes: Zu den angegebenen arrival times wird ein Task in die Ready Queue gestellt. Der Task kann frühestens beim nächsten diskreten Zeitpunkt dem Prozessor zugeteilt werden (Beispiel siehe Task A: arrival time= 0; Zuteilung = 1). Ein eintreffender Task wird immer ans Ende der Ready Queue gestellt (bei Feedback Scheduling immer ans Ende der höchst-prioren Queue).

Tragen Sie in der Zeile **Exec.** jenen Task ein der für die entsprechende Zeiteinheit dem Prozessor zugeteilt wird. Der restliche Raster stellt die Ready Queue dar. Tragen Sie hier jene Tasks ein die in der/den Ready Queue(s) stehen (beginnend in der obersten Zeile mit dem Task der als nächstes den Prozessor zugeteilt bekommt, usw., für Feedback-scheduling reihen Sie die höher prioren Queues zuerst). Schedulen Sie das Task Set nach der Round-Robin Methode und nach der Feedback Methode (time-slice = 1, die Anzahl der Queues für die Feedback Methode ist nicht beschränkt). Der Overhead für den Taskwechsel ist vernachlässigbar.

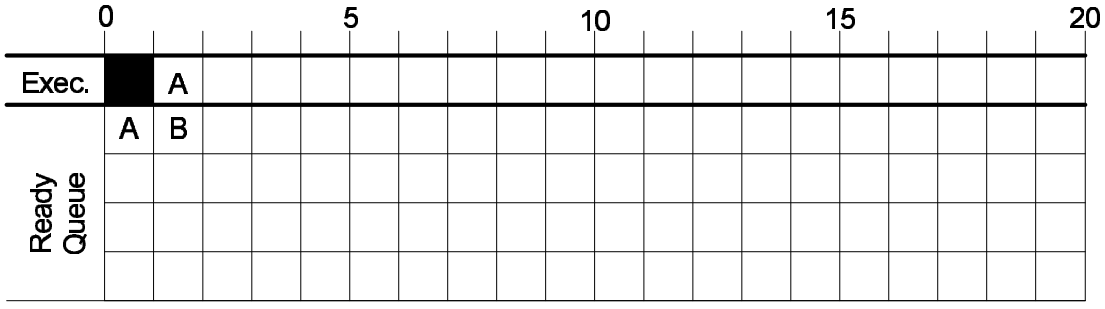


Abbildung 1: Round Robin Scheduling

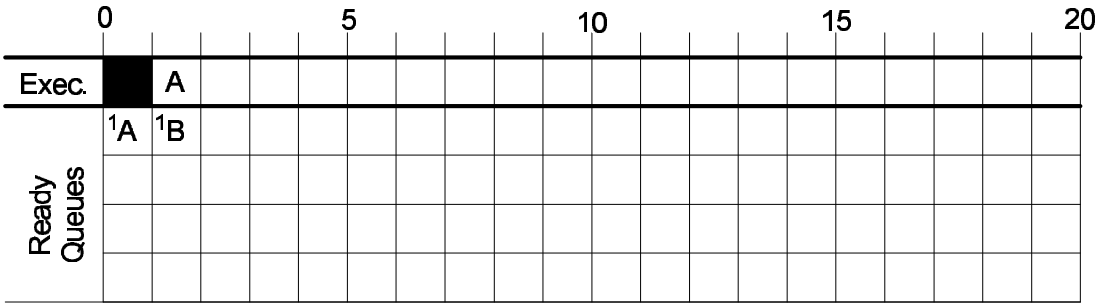


Abbildung 2: Feedback Scheduling

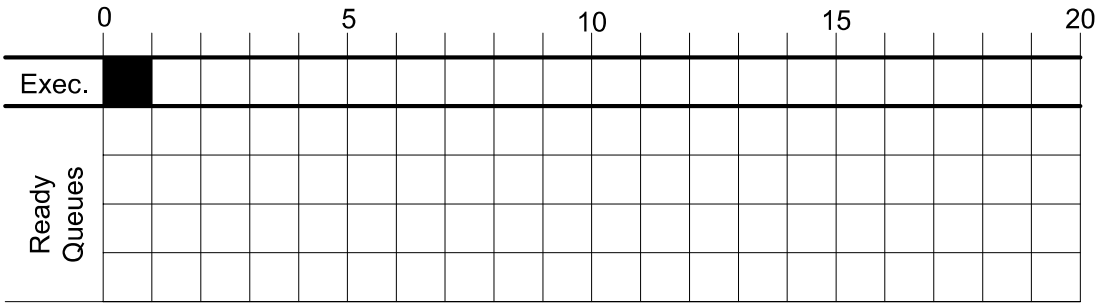


Abbildung 3: Ersatzraster

Rate-Monotonic Scheduling

Schedulen Sie das nebenstehende Taskset nach dem Rate-Monotonic Verfahren. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar. Gehen Sie davon aus, dass alle Tasks die erste Arrival Time 0 haben.

Task	Ausführungszeit	Periodendauer
A	1	4
B	3	10
C	2	5
D	1	16

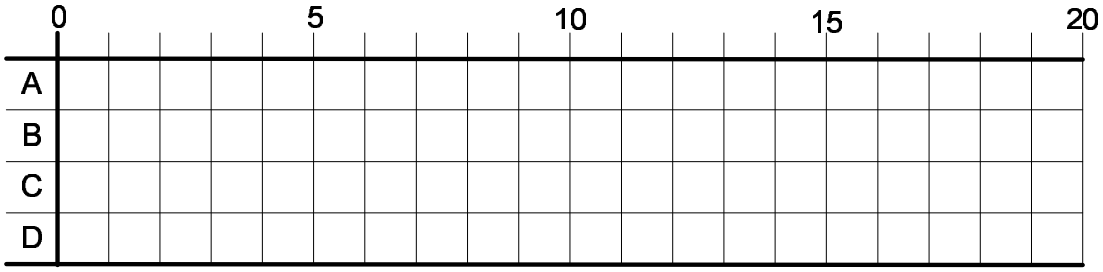


Abbildung 4: Rate-Monotonic Scheduling

2 Scheduling (25)

Priority Scheduling (8)

Schedulen Sie das nebenstehende Task Set. Beachten Sie dabei folgendes: Zu den angegebenen Arrival Times wird ein Task in die der Priorität des Tasks entsprechende Priority Queue gestellt. Jeder Task kann frühestens beim nächsten diskreten Zeitpunkt dem Prozessor zugeteilt werden (Beispiel siehe Task A: Arrival Time= 0; Zuteilung = 1). Ein eintreffender Task wird immer ans Ende der entsprechenden Priority Queue gestellt. Abbildung 1 zeigt das Abarbeitungsschema.

Task	Arrival Time	Service Time	Priority
A	0	2	1
B	1	3	2
C	1	1	3
D	4	1	2
E	4	1	1
F	6	2	3
G	6	1	2
H	7	1	3
I	10	2	1
K	11	3	3

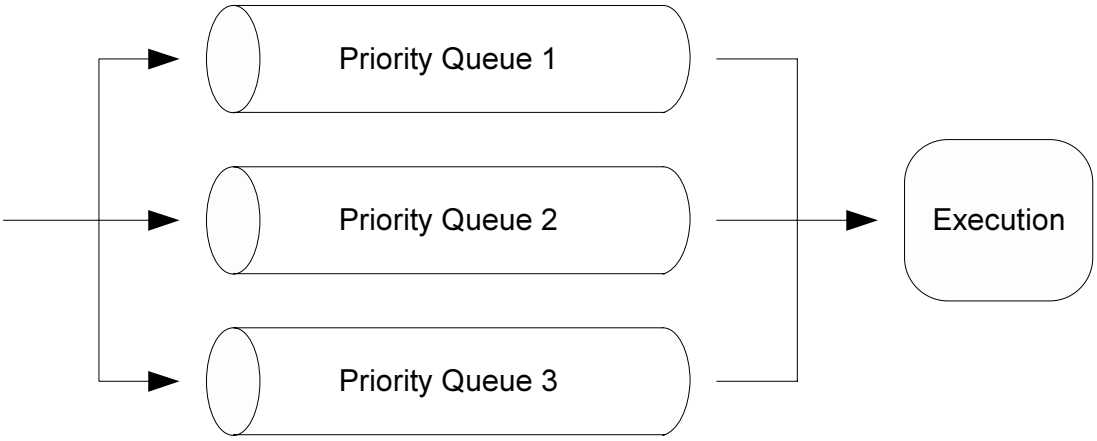
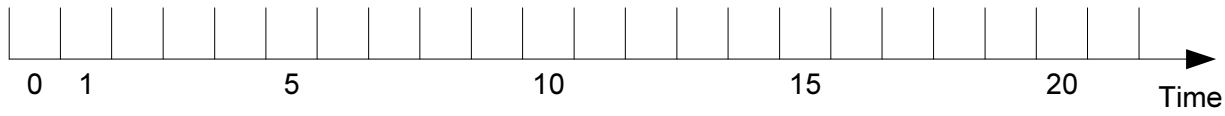


Abbildung 1: Priority Queuing

Beachten Sie, dass die Abarbeitung der Tasks *non preemptive* ist. Schreiben Sie in Abbildung 1 über jeder Priority Queue jene Tasks in zeitlich korrekter Reihenfolge, die dieser Queue zugewiesen werden. Tragen Sie weiters in das folgende Zeitschema die Abarbeitungsreihenfolge der Tasks ein. Der Overhead für den Taskwechsel ist vernachlässigbar.



Verständnisfrage (Eine fehlende Antwort wird negativ, eine falsche Antwort wird doppelt negativ gewertet!)

◯ Ja ◯ Nein Kann es beim Scheduling mit Priority Queues zur Starvation kommen?

Starvation (3)

Erklären Sie den Begriff *Starvation*:

Round-Robin Scheduling (7)

Task	Arrival Time	Service Time
A	0	3
B	5	5
C	7	2
D	10	4
E	15	2

Schedulen Sie das nebenstehende Task Set. Beachten Sie dabei folgendes: Zu den angegebenen Arrival Times wird ein Task in die Ready Queue gestellt. Der Task kann frühestens beim nächsten diskreten Zeitpunkt dem Prozessor zugeteilt werden (Beispiel siehe Task A: Arrival Time= 0; Zuteilung = 1). Ein eintreffender Task wird immer ans Ende der Ready Queue gestellt.

Tragen Sie in der Zeile **P** jenen Task ein, der für die entsprechende Zeiteinheit dem Prozessor zugeteilt wird. Der restliche Raster stellt die Ready Queue dar. Tragen Sie hier jene Tasks ein die in der Ready Queue stehen (beginnend in der obersten Zeile mit dem Taks, der als nächstes den Prozessor zugeteilt bekommt). Schedulen Sie das Task Set nach der Round-Robin Methode mit time-slice = 1. Der Overhead für den Taskwechsel ist vernachlässigbar.

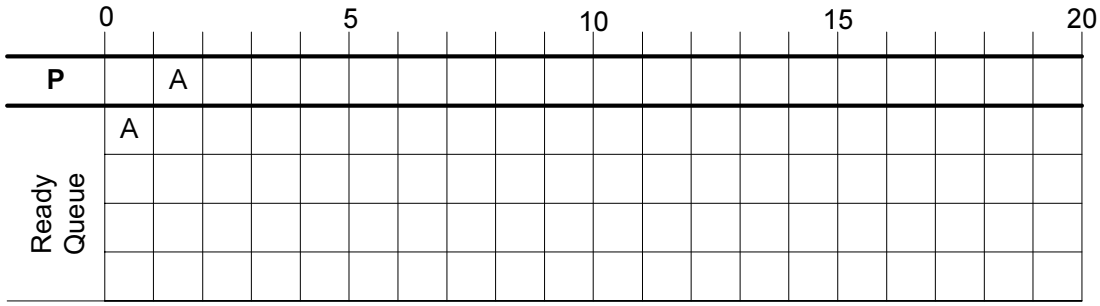


Abbildung 2: Round Robin

Verständnisfragen (7)

Beurteilen Sie die folgenden Aussagen! Fehlende Antworten werden negativ, falsche Antworten werden doppelt negativ gewertet!

◯ Ja ◯ Nein Die Prioritäten beim EDF Scheduling ergeben sich durch die Periodendauer der Tasks.

◯ Ja ◯ Nein Die Prioritäten beim RMS Scheduling ergeben sich durch die *Processor Utilization* der Tasks.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Single-Prozessor Systemen immer eine Lösung.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Single-Prozessor Systemen immer eine Lösung, wenn eine solche existiert.

◯ Ja ◯ Nein Bei Round Robin Scheduling kann das Problem der Starvation auftreten.

◯ Ja ◯ Nein Die First-Come-First-Serve-Strategie begünstigt Prozesse mit kurzer Ausführungszeit.

◯ Ja ◯ Nein Das Round-Robin-Verfahren ist preemptive.

2 Scheduling (25)

2.1 Round-Robin Scheduling 10

Schedulen Sie das folgende Taskset mittels Round-Robin und einer Zeitscheibenlänge (time-slice) gleich 1. Der Overhead für den Taskwechsel ist vernachlässigbar.

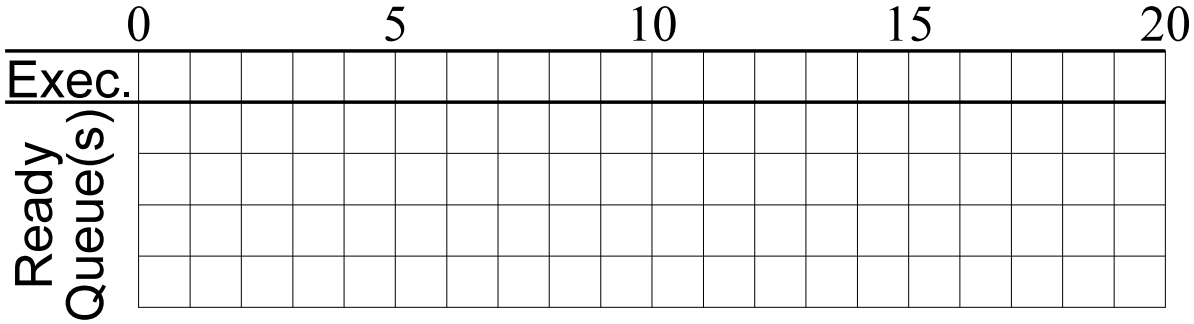
Das Scheduling soll in **folgender Reihenfolge** ablaufen (Algorithmus):

1. Neue Tasks an das Ende der Ready Queue stellen
2. Den zuletzt ausgeführten Task an das Ende der Ready Queue stellen
3. Den vordersten Task der Ready Queue ausführen (`Exec.`)

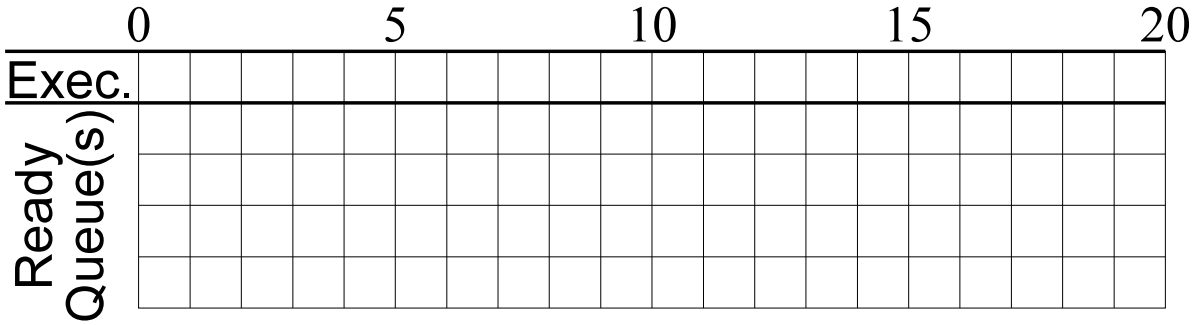
Task	Arrival Time	Execution Time
A	0	3
B	1	6
C	3	3
D	5	5
E	9	1
F	15	2

Ein Task kann also zum Zeitpunkt des Arrivals bereits ausgeführt werden (Beispiel siehe Task A: arrival time= 0; Zuteilung = 0).

Tragen Sie in der Zeile **Exec.** jenen Task ein der für die entsprechende Zeiteinheit dem Prozessor zugeteilt wird. Der restliche Raster stellt die Ready Queue dar. Tragen Sie hier jene Tasks ein die in der/den Ready Queue(s) stehen (beginnend in der obersten Zeile mit dem Task der als nächstes den Prozessor zugeteilt bekommt, usw.



Ersatzvorlage:



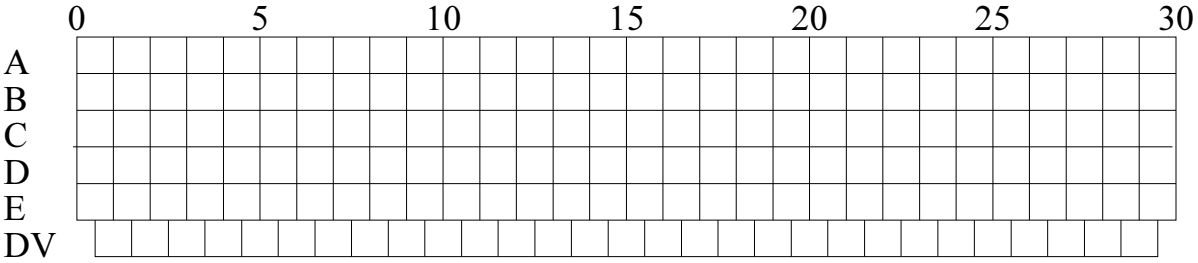
2.2 Earliest Deadline First Scheduling 12

Schedulen Sie das nebenstehende Taskset mittels Earliest Deadline First (EDF). Der Overhead für den Taskwechsel ist vernachlässigbar.

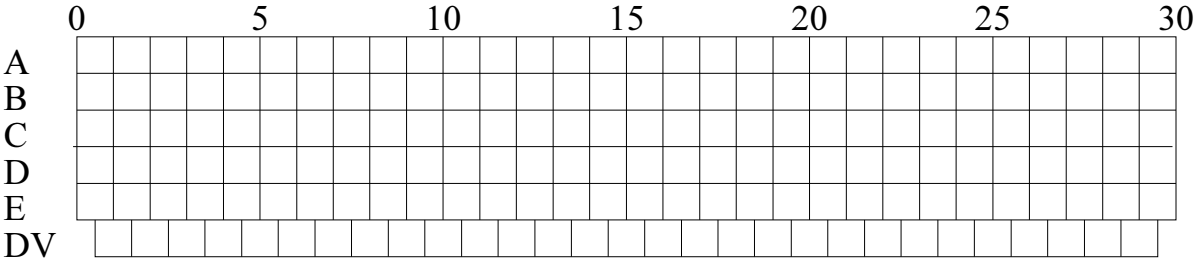
Task	Arrival Time (ms)	Execution Time (ms)	Deadline (ms)
A	0	10	20
B	2	3	10
C	3	4	9
D	6	1	7
E	16	3	19

Ein Task kann bereits zum Zeitpunkt des Arrivals ausgeführt werden (Beispiel siehe Task A: arrival time = 0; Zuteilung = 0).

Verfollständigen Sie folgendes Diagramm bis alle Tasks gescheduled worden sind bzw. bis es zu einer Deadline-Verletzung kommt. Tragen Sie im Falle einer Deadline-Verletzung zum betreffenden Zeitpunkt den jeweiligen Task in der Zeile `DV` ein.



Ersatzvorlage:



Bitte beantworten Sie diese Frage unabhängig von Ihrer Lösung oberhalb: Nehmen Sie an, dass das obige Taskset gescheduled werden kann. Bis zu welchem Zeitpunkt müssen Sie das angegebene Taskset *mindestens* schedulen, um sicher gehen zu können, dass es wirklich schedulbar ist. Bitte richtige Antworten ankreuzen:

◯ 10 ms ◯ 16 ms ◯ 18 ms ◯ 19 ms ◯ 20 ms ◯ 21 ms ◯ 22 ms
◯ 23 ms ◯ 38 ms ◯ einen Zeitpunkt der in der Liste nicht aufscheint

2.3 Allgemeine Fragen zu Scheduling 3

Bewertung: falsche Antwort: -2, keine Antwort: -1

Die Verwendung von Preemtion bei Scheduling erhöht den Durchsatz des Systems:

◯ korrekt ◯ inkorrekt

Die Verwendung von First-Come-First-Served (FCFS) kann im ungünstigen Fall zu Starvation führen:

◯ korrekt ◯ inkorrekt

Die Verwendung von Round-Robin kann es zu keiner Starvation kommen:

◯ korrekt ◯ inkorrekt

VO 182.022 18. Jänner 2005

Prüfung Betriebssysteme

KNr. MNr. Zuname, Vorname

Ges.)(100) 1.)(25) 2.)(25) 3.)(25) 4.)(25) Zusatzblätter:

Bitte verwenden Sie nur dokumentenechtes Schreibmaterial!

1 Scheduling (25)

1.1 Uniprocessor Scheduling (13)

Gegeben ist nebenstehendes Taskset. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar.

Task	Ausführungszeit	Periodendauer
A	1	5
B	2	10
C	2	7
D	1	6

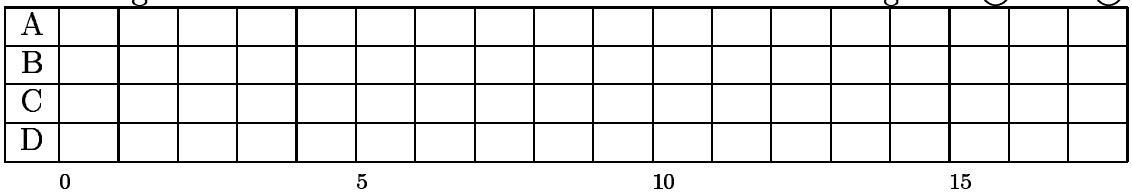
Ermitteln Sie für dieses Taskset die *notwendige* und die *hinreichende* Bedingung für das *Rate Monotonic Scheduling* (RMS) Verfahren. Berechnen Sie die **Zahlenwerte** überschlagsmäßig (ohne Taschenrechner)!

Ist die notwendige Bedingung erfüllt? ◯ Ja ◯ Nein

Ist die hinreichende Bedingung erfüllt? ◯ Ja ◯ Nein

Versuchen Sie das Taskset einmal mit dem RMS und einmal mit dem *Earliest-Deadline-First* (EDF) Verfahren zu schedulen. Verwenden Sie dazu die nachstehenden Vorlagen. Tragen Sie bei jeder Vorlage die aktiven Taskzeiten ein und bezeichnen Sie deutlich eventuelle Deadlineverletzungen. Kreuzen Sie jeweils an ob das Scheduling erfolgreich war. Eine Vorlage dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

Scheduling nach dem **RMS**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Scheduling nach dem **EDF**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein

A																	
B																	
C																	
D																	
	0				5					10					15		

Ersatzvorlage: Scheduling nach dem -Verfahren: Erfolgreich: ◯ Ja ◯ Nein

A																	
B																	
C																	
D																	
	0				5					10					15		

1.2 Verständnisfragen (12)

Beurteilen Sie die folgenden Aussagen für Single-Prozessor Scheduling!
Fehlende Antworten werden negativ, falsche Antworten werden doppelt negativ gewertet!

◯ Ja ◯ Nein Jedes Taskset, das mittels EDF gelöst werden kann, lässt auch mit RMS schedulen.

◯ Ja ◯ Nein Ein Taskset, das mittels EDF nicht gelöst werden kann, lässt eventuell mit RMS schedulen.

◯ Ja ◯ Nein Die Prioritäten beim EDF Scheduling ergeben sich durch die Periodendauer der Tasks.

◯ Ja ◯ Nein Bei Round Robin Scheduling kann das Problem der Starvation auftreten.

◯ Ja ◯ Nein Beim Scheduling nach dem Round-Robin-Verfahren werden I/O-intensive Prozesse benachteiligt.

◯ Ja ◯ Nein Wenn bei allen Tasks die Deadline gleich ihrer Periode ist, dann liefert RMS Scheduling dasselbe Ergebnis wie EDF Scheduling.

◯ Ja ◯ Nein Die First-Come-First-Serve-Strategie benachteiligt CPU-intensive Prozesse.

◯ Ja ◯ Nein Das Shortest-Remaining-Time-Verfahren begünstigt I/O-intensive Prozesse.

◯ Ja ◯ Nein Das Shortest-Process-Next-Verfahren ist für Echtzeitscheduling ungeeignet.

◯ Ja ◯ Nein Die Prioritäten beim RMS Scheduling ergeben sich durch die reziproke Periodendauer der Tasks.

◯ Ja ◯ Nein Das Shortest-Process-Next-Verfahren kommt im Vergleich zu Round Robin mit weniger Interruptaufrufen aus.

2 Scheduling (25)

Rate Monotonic Scheduling (9)

Schedulen Sie das nebenstehende Taskset nach dem Rate-Monotonic Verfahren. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar. Gehen Sie davon aus, dass alle Tasks die erste Arrival Time 0 haben.

Task	Ausführungszeit	Periodendauer
A	1	4
B	2	15
C	1	9
D	3	11
E	2	13

Tragen Sie in der Zeile **P** jenen Task ein, der für die entsprechende Zeiteinheit dem Prozessor zugeteilt wird.

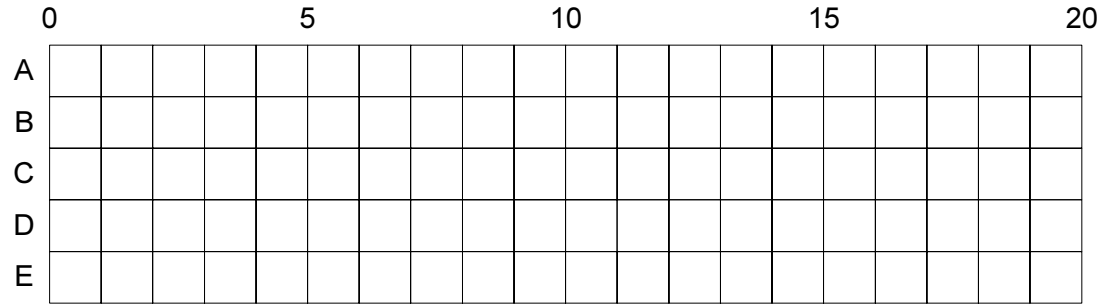


Abbildung 1: Round Robin

Kann es bei diesem Taskset zu einem Deadline-Miss kommen? Begründen Sie Ihre Antwort!

Round-Robin (8)

Schedulen Sie das nebenstehende Taskset. Beachten Sie dabei folgendes: Zu den angegebenen arrival times wird ein Task in die Ready Queue gestellt. Der Task kann frühestens beim nächsten diskreten Zeitpunkt dem Prozessor zugeteilt werden (Beispiel siehe Task A: arrival time= 0; Zuteilung = 1). Ein eintreffender Task wird immer ans Ende der Ready Queue gestellt.

Task	Arrival Time	Service Time
A	0	5
B	1	3
C	4	2
D	5	7
E	7	2

Tragen Sie in der Zeile **Exec.** jenen Task ein der für die entsprechende Zeiteinheit dem Prozessor zugeteilt wird. Der restliche Raster stellt die Ready Queue dar. Tragen Sie hier jene Tasks ein die in der/den Ready Queue(s) stehen (beginnend in der obersten Zeile mit

dem Task der als nächstes den Prozessor zugeteilt bekommt. Schedulen Sie das Task Set nach der Round-Robin Methode (time-slice = 1). Der Overhead für den Taskwechsel ist vernachlässigbar.

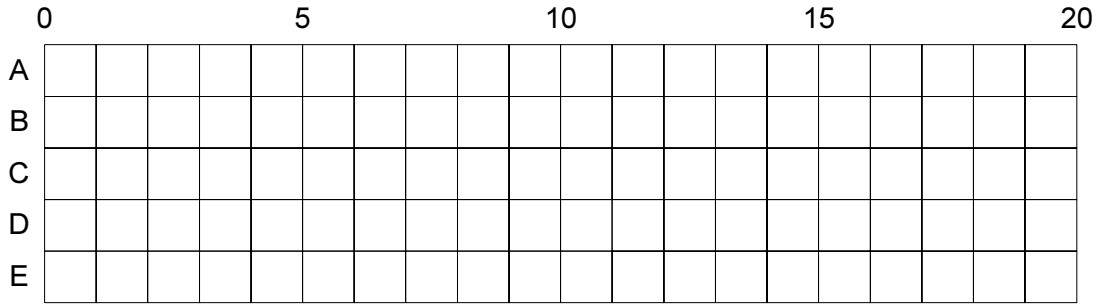


Abbildung 2: Round Robin Scheduling

Verständnisfragen (8)

Was bedeutet Starvation?

Geben Sie 3 Scheduling Methoden an bei denen es zu Starvation kommen kann:

Erklären Sie das **gemeinsame** Problem dieser 3 Methoden, wieso es zu Starvation kommen kann:

2 Scheduling (25)

2.1 Uniprocessor Scheduling (13)

Gegeben ist nebenstehendes Taskset. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar.

Task	Ausführungszeit	Periodendauer
A	1	10
B	1	5
C	1	4
D	2	7

Ermitteln Sie für dieses Taskset die *notwendige* und die *hinreichende* Bedingung für das *Rate Monotonic Scheduling* (RMS) Verfahren. Berechnen Sie die **Zahlenwerte** überschlagsmäßig (ohne Taschenrechner)!

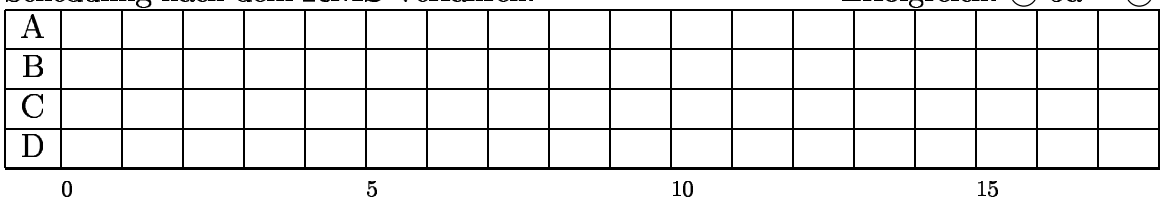
Benutzen Sie dazu folgende Funktionstabelle:

n	1	2	3	4	5	6
$n \cdot (2^{(1/n)} - 1)$	1.00	0.82	0.77	0.75	0.74	0.73

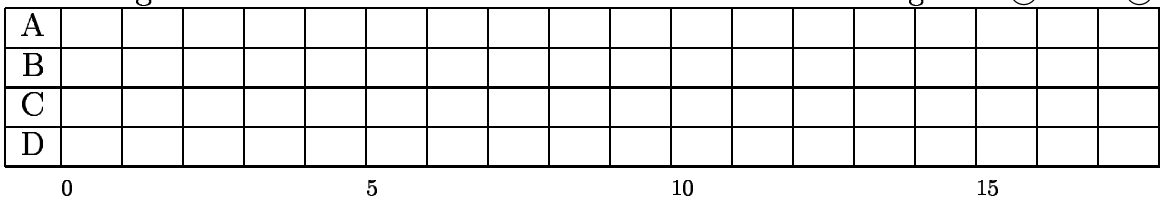
Ist die notwendige Bedingung erfüllt? ◯ Ja ◯ Nein
Ist die hinreichende Bedingung erfüllt? ◯ Ja ◯ Nein

Versuchen Sie das Taskset einmal mit dem RMS und einmal mit dem *Earliest-Deadline-First* (EDF) Verfahren zu schedulen. Verwenden Sie dazu die nachstehenden Vorlagen. Tragen Sie bei jeder Vorlage die aktiven Taskzeiten ein und bezeichnen Sie deutlich eventuelle Deadlineverletzungen. Kreuzen Sie jeweils an ob das Scheduling erfolgreich war. Eine Vorlage dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

Scheduling nach dem **RMS**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Scheduling nach dem **EDF**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Ersatzvorlage: Scheduling nach dem [redacted]-Verfahren: Erfolgreich: ◯ Ja ◯ Nein

A																		
B																		
C																		
D																		
	0					5					10					15		

2.2 Verständnisfragen (12)

Beurteilen Sie die folgenden Aussagen für Single-Prozessor Scheduling!
Fehlende Antworten werden negativ, falsche Antworten werden doppelt negativ gewertet!

◯ Ja ◯ Nein Die Prioritäten beim EDF Scheduling ergeben sich durch die Periodendauer der Tasks.

◯ Ja ◯ Nein Die Prioritäten beim RMS Scheduling ergeben sich durch die *Processor Utilization* der Tasks.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Single-Prozessor Systemen immer eine Lösung.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Multi-Prozessor Systemen immer eine Lösung, wenn eine solche existiert.

◯ Ja ◯ Nein Das EDF Scheduling ist non-preemptive.

◯ Ja ◯ Nein Bei Round Robin Scheduling kann das Problem der Starvation auftreten.

◯ Ja ◯ Nein Beim Scheduling nach dem Round-Robin-Verfahren werden I/O-intensive Prozesse benachteiligt.

◯ Ja ◯ Nein Die First-Come-First-Serve-Strategie benachteiligt CPU-intensive Prozesse.

◯ Ja ◯ Nein Das Round-Robin-Verfahren ist preemptive.

◯ Ja ◯ Nein Das Shortest-Process-Next-Verfahren ist für Echtzeitscheduling ungeeignet.

◯ Ja ◯ Nein Das Shortest-Process-Next-Verfahren kommt im Vergleich zu Round Robin mit weniger Interruptaufrufen aus.

◯ Ja ◯ Nein Ein Scheduler nach dem RMS Verfahren benötigt mehr Information pro Task als ein Scheduler nach dem Round-Robin-Verfahren.

2 Uniprocessor Scheduling (20)

Gegeben ist nebenstehendes Taskset. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar.

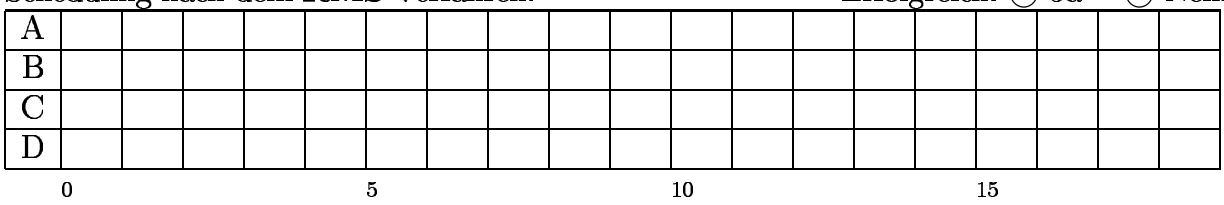
Task	Ausführungszeit	Periodendauer
A	2	4
B	1	6
C	1	7
D	1	9

Ermitteln Sie für dieses Taskset die *notwendige* und die *hinreichende* Bedingung für das *Rate Monotonic Scheduling* (RMS) Verfahren. Berechnen Sie die **Zahlenwerte** überschlagsmäßig (ohne Taschenrechner)!

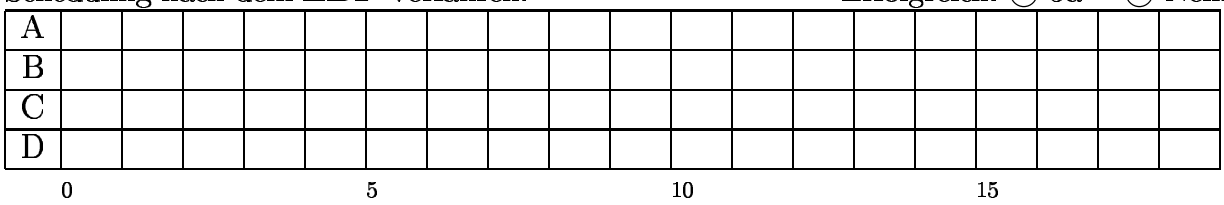
Ist die notwendige Bedingung erfüllt? ◯ Ja ◯ Nein
Ist die hinreichende Bedingung erfüllt? ◯ Ja ◯ Nein

Versuchen Sie das Taskset einmal mit dem RMS und einmal mit dem *Earliest-Deadline-First* (EDF) Verfahren zu schedulen. Verwenden Sie dazu die nachstehenden Vorlagen. Tragen Sie bei jeder Vorlage die aktiven Taskzeiten ein und bezeichnen Sie deutlich eventuelle Deadlineverletzungen. Kreuzen Sie jeweils an, ob das Scheduling erfolgreich war. Eine Vorlage dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

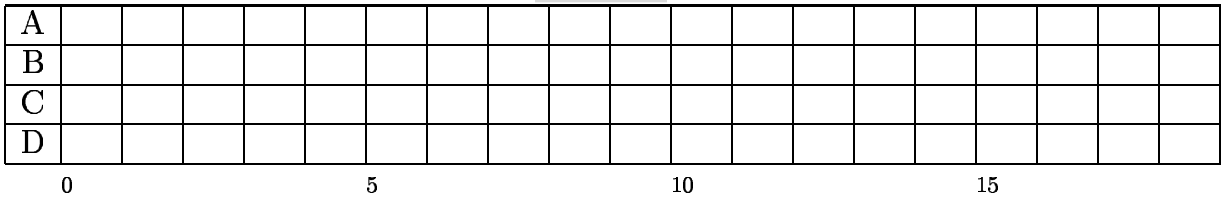
Scheduling nach dem **RMS**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Scheduling nach dem **EDF**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Ersatzvorlage: Scheduling nach dem -Verfahren: Erfolgreich: ◯ Ja ◯ Nein



2 Uniprocessor Scheduling (20)

Gegeben ist nebenstehendes Taskset. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar.

Task	Ausführungszeit	Periodendauer
A	1	7
B	1	6
C	2	4
D	1	9

Ermitteln Sie für dieses Taskset die *notwendige* und die *hinreichende* Bedingung für das *Rate Monotonic Scheduling* (RMS) Verfahren. Berechnen Sie die **Zahlenwerte** überschlagsmäßig ($\sqrt[2]{2} \approx 1,41$ $\sqrt[3]{2} \approx 1,26$ $\sqrt[4]{2} \approx 1,19$ $\sqrt[5]{2} \approx 1,15$ $\sqrt[6]{2} \approx 1,12$).

Ist die notwendige Bedingung erfüllt? ◯ Ja ◯ Nein
Ist die hinreichende Bedingung erfüllt? ◯ Ja ◯ Nein

Versuchen Sie das Taskset einmal mit dem RMS und einmal mit dem *Earliest-Deadline-First* (EDF) Verfahren zu schedulen. Verwenden Sie dazu die nachstehenden Vorlagen. Tragen Sie bei jeder Vorlage die aktiven Taskzeiten ein und bezeichnen Sie deutlich eventuelle Deadlineverletzungen. Kreuzen Sie jeweils an, ob das Scheduling erfolgreich war. Eine Vorlage dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

Scheduling nach dem **RMS**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein

A																		
B																		
C																		
D																		
	0					5					10					15		

Scheduling nach dem **EDF**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein

A																		
B																		
C																		
D																		
	0					5					10					15		

Ersatzvorlage: Scheduling nach dem -Verfahren: Erfolgreich: ◯ Ja ◯ Nein

A																		
B																		
C																		
D																		
	0					5					10					15		

3 Uniprocessor Scheduling (20)

Gegeben ist nebenstehendes Taskset. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar.

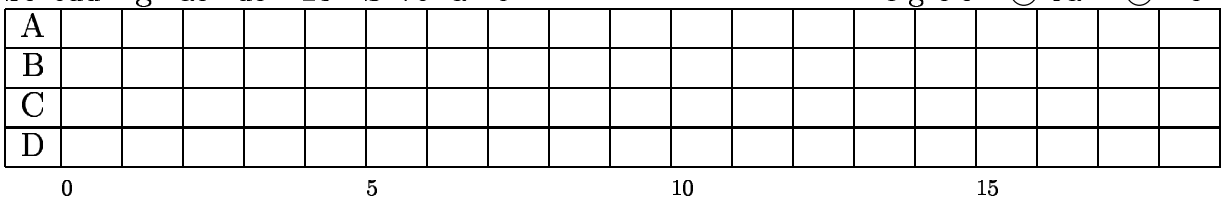
Task	Ausführungszeit	Periodendauer
A	2	8
B	1	4
C	2	9
D	1	5

Ermitteln Sie für dieses Taskset die *notwendige* und die *hinreichende* Bedingung für das *Rate Monotonic Scheduling* (RMS) Verfahren. Berechnen Sie die **Zahlenwerte** überschlagsmäßig ($\sqrt[2]{2} \approx 1,41 \quad \sqrt[3]{2} \approx 1,26 \quad \sqrt[4]{2} \approx 1,19 \quad \sqrt[5]{2} \approx 1,15 \quad \sqrt[6]{2} \approx 1,12$).

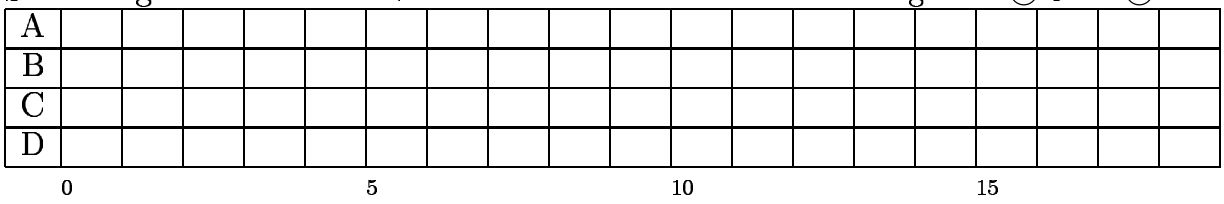
Ist die notwendige Bedingung erfüllt? ◯ Ja ◯ Nein
Ist die hinreichende Bedingung erfüllt? ◯ Ja ◯ Nein

Versuchen Sie das Taskset einmal mit dem RMS und einmal mit dem *Earliest-Deadline-First* (EDF) Verfahren zu schedulen. Verwenden Sie dazu die nachstehenden Vorlagen. Tragen Sie bei jeder Vorlage die aktiven Taskzeiten ein und bezeichnen Sie deutlich eventuelle Deadlineverletzungen. Kreuzen Sie jeweils an, ob das Scheduling erfolgreich war. Eine Vorlage dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

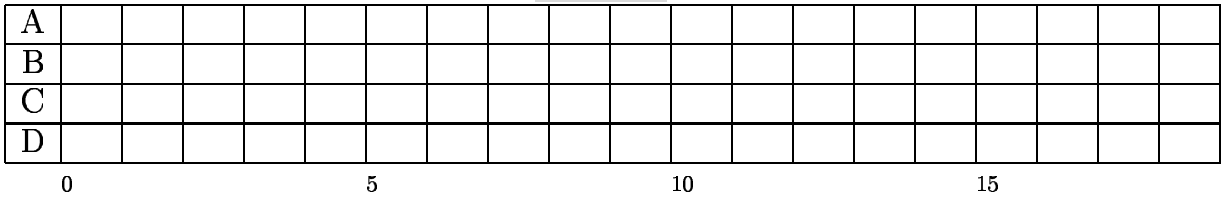
Scheduling nach dem **RMS**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Scheduling nach dem **EDF**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Ersatzvorlage: Scheduling nach dem -Verfahren: Erfolgreich: ◯ Ja ◯ Nein



4 Memory Management (32)

a) Virtueller Speicher mit Kombination aus Segmentierung und Paging 18

In folgendem Beispiel sollen Sie ausgehend von virtuellen Adressen die physikalischen Adressen bestimmen. Es werden folgende Begriffe (englische Notation) aus dem Buch zur Vorlesung verwendet:

Base	Basisadresse Seitentabelle des Segmentes
Length	Länge des Segmentes (Anzahl der Seiten des Segmentes)
Virt.Addr.	Virtuelle Adresse
Frame#	Seitenrahmennummer (im physischen Speicher)
Page#	Seitennummer (im virtuellen Speicher)
Seg#	Segmentnummer

Es handelt sich bei dem System um ein System mit 32-Bit-Adressen, wobei die niederwertigen 16 Bit immer den Offset einer Adresse bilden und die höherwertigen 16 Bit Segment- und Seitennummer bilden:

Seg# (8 bit)	Page# (8 bit)	Offset (16 bit)

Es wird dabei direkter Zugriff (direct mapping) sowohl auf die Segmenttabelle als auch auf die Seitentabelle verwendet.

Bei Paging sind alle Seiten 65536 Bytes (hexadezimal `0x0001 0000`) lang. Alle Werte sind als Hexadezimalzahlen angegeben. Ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte "invalid segment nr." bzw. "invalid page nr." in das entsprechende Feld.

Verwenden Sie für die Adressumsetzung folgende Segmenttabelle:

Segmenttabelle		
Seg#	Base	Length
0x00	0x0102 7234	0x03
0x01	0x0300 6073	0x01
0x02	0xC9C0 8054	0x02
0x03	0x8A9F 23AA	0x10
0x04	0x2001 B578	0x0A

und folgende Seitentabelle:

Seitentabelle	
Address	Frame#
...	...
0x0102 7234	0x6752
0x0102 7235	0x7613
0x0102 7236	0x258A
0x0102 7237	0xB3CA
...	...
0x0300 6073	0x56EF
0x0300 6074	0x4567
...	...
0x2001 B57F	0x5014
0x2001 B580	0x5109
0x2001 B581	0xA145
0x2001 B582	0x2000
0x2001 B583	0x3234
...	...
0xC9C0 8054	0x7353
0xC9C0 8055	0xBABA
0xC9C0 8056	0x9789
0xC9C0 8057	0xAFFE
...	...
0x8A9F 23B7	0x5400
0x8A9F 23B8	0x5401
0x8A9F 23B9	0x0945
0x8A9F 23BA	0x1313
...	...

Ermitteln Sie unter Benützung obiger Tabellen die physikalischen Adressen zu den folgenden virtuellen Adressen. Markieren Sie die den Eintrag mit "invalid segment nr." bzw. "invalid page nr." im Fehlerfall.

Virtuelle Adresse	Physikalische Adresse (zu ermitteln)
0x030F 01CE	
0x0001 04B3	
0x8A9F 23B9	
0x0409 1025	
0x0409 0102	
0x04B0 1019	
0x0539 0715	
0x0407 294A	
0x0310 9728	

Bitte beachten Sie, dass bei diesem Beispiel falsche Antworten zu Punkteabzug führen.

b) Verständnisfragen (14)

Kreuzen Sie bitte die richtigen Antworten an! Achtung! Falsche Antworten werden negativ gewertet!

3 Uniprocessor Scheduling (20)

Gegeben ist nebenstehendes Taskset. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar.

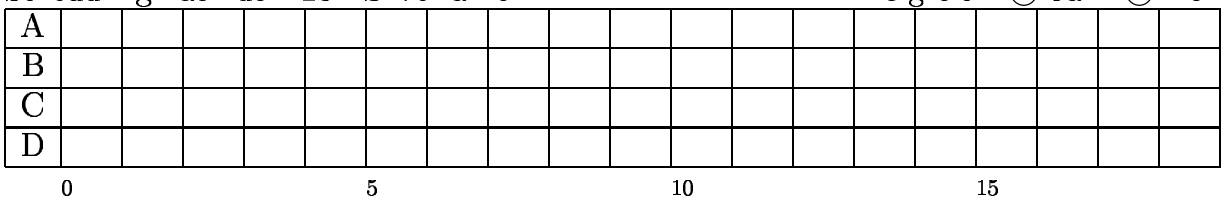
Task	Ausführungszeit	Periodendauer
A	1	7
B	2	6
C	2	8
D	1	9

Ermitteln Sie für dieses Taskset die *notwendige* und die *hinreichende* Bedingung für das *Rate Monotonic Scheduling* (RMS) Verfahren. Berechnen Sie die **Zahlenwerte** überschlagsmäßig ($\sqrt[2]{2} \approx 1,41 \quad \sqrt[3]{2} \approx 1,26 \quad \sqrt[4]{2} \approx 1,19 \quad \sqrt[5]{2} \approx 1,15 \quad \sqrt[6]{2} \approx 1,12$).

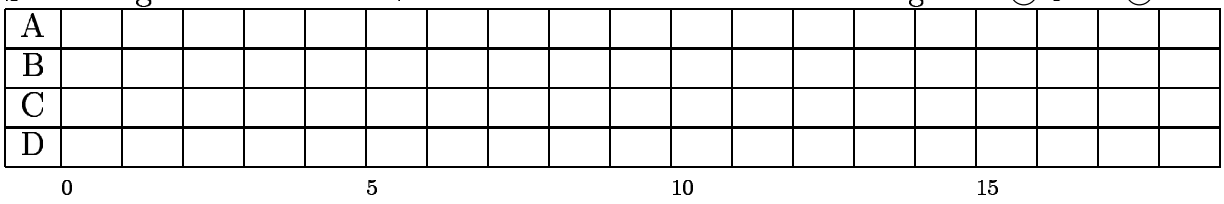
Ist die notwendige Bedingung erfüllt? ◯ Ja ◯ Nein
Ist die hinreichende Bedingung erfüllt? ◯ Ja ◯ Nein

Versuchen Sie das Taskset einmal mit dem RMS und einmal mit dem *Earliest-Deadline-First* (EDF) Verfahren zu schedulen. Verwenden Sie dazu die nachstehenden Vorlagen. Tragen Sie bei jeder Vorlage die aktiven Taskzeiten ein und bezeichnen Sie deutlich eventuelle Deadlineverletzungen. Kreuzen Sie jeweils an, ob das Scheduling erfolgreich war. Eine Vorlage dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

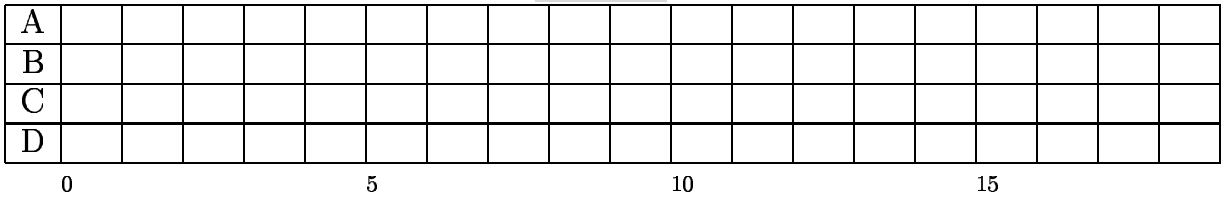
Scheduling nach dem **RMS**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Scheduling nach dem **EDF**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Ersatzvorlage: Scheduling nach dem -Verfahren: Erfolgreich: ◯ Ja ◯ Nein



2 Scheduling (25)

2.1 Uniprocessor Rate Monotonic Scheduling (10)

Gegeben ist nebenstehendes Taskset. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar.

Task	Ausführungszeit	Periodendauer
A	2	7
B	2	6
C	1	9
D	1	7

Ermitteln Sie für dieses Taskset die *notwendige* und die *hinreichende* Bedingung für das *Rate Monotonic Scheduling* (RMS) Verfahren. Berechnen Sie die **Zahlenwerte** überschlagsmässig ($\sqrt[2]{2} \approx 1,41 \quad \sqrt[3]{2} \approx 1,26 \quad \sqrt[4]{2} \approx 1,19 \quad \sqrt[5]{2} \approx 1,15 \quad \sqrt[6]{2} \approx 1,12$).

[redacted]

Ist die notwendige Bedingung erfüllt? ◯ Ja ◯ Nein
Ist die hinreichende Bedingung erfüllt? ◯ Ja ◯ Nein

Versuchen Sie das Taskset mit dem RMS Verfahren zu schedulen. Verwenden Sie dazu die nachstehenden Vorlagen. Tragen Sie bei jeder Vorlage die aktiven Taskzeiten ein und bezeichnen Sie deutlich eventuelle Deadlineverletzungen. Kreuzen Sie an, ob das Scheduling erfolgreich war. Eine Vorlage dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

Scheduling nach dem **RMS**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein

A																		
B																		
C																		
D																		

0 5 10 15

Ersatzvorlage: Scheduling nach dem RMS-Verfahren: Erfolgreich: ◯ Ja ◯ Nein

A																		
B																		
C																		
D																		

0 5 10 15

2.2 Uniprocessor Round Robin Scheduling (7)

Scheduling nach dem Round Robin Verfahren: Versuchen Sie das Taskset nach dem *Round Robin* (RR) Verfahren zu schedulen. Fügen Sie zuerst *Tasks mit abgelaufener Zeitscheibe vor neu ankommenden Tasks* in die Ready Queue. Verwenden Sie die nachstehenden Vorlagen. Tragen Sie in die Vorlage die aktiven Taskzeiten ein und bezeichnen Sie deutlich (mit einem Pfeil →) die Arrival Time der Tasks. Eine Zeitscheibe entspricht einer *Service Time* von eins. Eine der Vorlagen dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

Process	Arrival Time	Service Time
P1	0	5
P2	1	7
P3	3	4
P4	6	2
P5	10	1

Scheduling nach dem **RR**-Verfahren:

P1																			
P2																			
P3																			
P4																			
P5																			
	0					5					10					15			

Ersatzvorlage: Scheduling nach dem RR-Verfahren:

P1																			
P2																			
P3																			
P4																			
P5																			
	0					5					10					15			

2.3 Scheduling allgemein (8)

Welche Scheduling–Kriterien kennen Sie? Vervollständigen Sie nachstehende Tabelle.

	User-oriented	System-Oriented
Performance	[redacted]	[redacted]
Other	[redacted]	[redacted]

2 Scheduling (25)

RR und SRT Scheduling (10)

Schedulen Sie das nebenstehende Taskset mit den Verfahren *Round Robin* und *Shortest Remaining Time* (SRT). Der Overhead für den Taskwechsel ist vernachlässigbar.

Task	Arrival Time	Service Time
P1	0	5
P2	2	2
P3	3	6
P4	8	4
P5	10	2

Beim Round Robin Verfahren soll die *Zeitscheibenlänge* 2 verwendet werden. Fügen Sie zuerst Tasks mit abgelaufener Zeitscheibe vor neu ankommenden Tasks in die Ready Queue ein. Wird ein Task beendet, bevor seine Zeitscheibe abgelaufen ist, wird sofort einem neuen Task die Ressource für 2 Zeiteinheiten zugewiesen.

Beim *SRT* Verfahren ist zu beachten, dass Tasks zum jeweiligen Ankunftszeitpunkt sofort gescheduled werden können. Stehen mehrere Tasks mit der selben verbliebenen Service Time zur Auswahl, soll der Task mit der niedrigsten Nummer gewählt werden (also beispielsweise $P4$ vor $P5$).

Verwenden Sie die nachstehenden Vorlagen. Markieren Sie in die Vorlage zu jedem Zeitpunkt den jeweils aktiven Task. Bezeichnen Sie weiters deutlich die *arrival time* jedes Tasks (mit einem Pfeil $\rightarrow$) und den Zeitpunkt, wenn ein Task die gesamte benötigte *service time* konsumiert hat ($\leftarrow$). Eine der Vorlagen dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

Scheduling nach dem **RR**-Verfahren:

P1	$\rightarrow$										$\leftarrow$									
P2																				
P3																				
P4																				
P5																				
	0					5						10					15			

Scheduling nach dem **SRT**-Verfahren:

P1	$\rightarrow$								$\leftarrow$											
P2																				
P3																				
P4																				
P5																				
	0					5						10					15			

Ersatztabelle: Scheduling nach dem _____ -Verfahren:

P1	→																			
P2																				
P3																				
P4																				
P5																				
	0					5					10					15				

2.1 Real-Time Scheduling (9)

Gegeben ist nebenstehendes Taskset. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar.

Task	Ausführungszeit	Periodendauer
A	1	6
B	2	8
C	1	9
D	3	11
E	1	13

Versuchen Sie das Taskset mit dem RMS Verfahren zu schedulen. Verwenden Sie dazu die nachstehenden Vorlagen. Tragen Sie bei jeder Vorlage die aktiven Taskzeiten ein und bezeichnen Sie deutlich eventuelle Deadlineverletzungen. Kreuzen Sie weiters an, ob das Scheduling erfolgreich war. Eine Vorlage dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

Scheduling nach dem **RMS**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein

A																			
B																			
C																			
D																			
E																			
	0					5					10					15			

Ersatzvorlage: Scheduling nach dem **RMS**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein

A																			
B																			
C																			
D																			
E																			
	0					5					10					15			

Kann es bei diesem Taskset mit dem *EDF* (Earliest Deadline First) Verfahren zu einer Deadlineverletzung kommen? Begründen Sie ihre Antwort!

[redacted]

2.2 Verständnisfragen (6)

Werden beim Round-Robin IO-intensive oder CPU-intensive Prozesse benachteiligt? Beschreiben Sie eine Variante des Round-Robin Verfahrens, die dieses Problem zu umgehen zu versucht.

Was versteht man unter dem Begriff Starvation? Nennen Sie ein Scheduling Verfahren, bei dem es zu Starvation kommen kann.

Bei welchen der folgenden Scheduling Verfahren muss die Service Time der Tasks bekannt sein (beziehungsweise geschätzt werden)? Fehlende Antworten werden negativ, falsche Antworten werden doppelt negativ gewertet!

◯ Ja ◯ Nein Virtual Round Robin

◯ Ja ◯ Nein Earliest Deadline First

◯ Ja ◯ Nein Highest Response Ratio Next

◯ Ja ◯ Nein Rate Monotonic Scheduling

◯ Ja ◯ Nein Shortest Process Next

◯ Ja ◯ Nein Feedback Scheduling