Name:

Matrikelnummer:

Bearbeitungszeit: 120 min.

Aufgabe 1.

Zur Programmierung eines Brettspiels mit einer bestimmten Anzahl von Spielfeldern (fortlaufend nummeriert beginnend mit 1) ist bereits die Klasse `Spielfigur` mit dem Konstruktor

```
Spielfigur(int anzahlFelder)
```

und den Methoden

```
/** Liefert die aktuelle Position (=Feld) der Spielfigur */
int getFeld();

/** Setzt die Spielfigur auf das angegebene Feld. */
void setFeld(int feld);

/** Fuehrt mit der Spielfigur den naechsten Zug aus. */
void ziehe();
```

gegeben. Die Methode `ziehe()` bewegt die Spielfigur nach bestimmten Regeln auf ein neues Feld.

Schreiben Sie die Klasse `Brettspiel` mit dem Konstruktor

```
public Brettspiel(int anzahlFelder),
```

der ein Brettspiel mit der angegebenen Anzahl von Feldern anlegt. Weiters soll die Klasse `Brettspiel` über die Methode

```
public int spiele(int anzahlFiguren)
```

verfügen, die ein Spiel mit der angegebenen Anzahl von Spielfiguren durchführt (die Spielfiguren werden von 1 beginnend durchnummeriert). Zu Beginn befinden sich alle Spielfiguren auf dem Startfeld mit der Nummer 1. Die Methode soll solange reihum die Spielfiguren ziehen (mit der ersten beginnend), bis erstmals eine Spielfigur das Zielfeld (mit der Nummer `anzahlFelder`) erreicht. Die Nummer dieser siegreichen Spielfigur soll von der Methode zurückgeliefert werden. Wird eine Spielfigur auf ein Feld gezogen, auf dem sich bereits eine andere Spielfigur befindet, so wird die bereits vorhandene Spielfigur auf das Startfeld zurückgesetzt. (Auf dem Startfeld können sich mehrere Spielfiguren befinden.)

Aufgabe 2. Eine Anzahl von Aufträgen soll möglichst gleichmäßig auf Maschinen verteilt werden. Jeder Auftrag kann auf einer beliebigen der Maschinen gefertigt werden, allerdings muss ein Auftrag immer zur Gänze auf der gewählten Maschine gefertigt werden. Die Aufträge sind von unterschiedlicher Größe. Für eine gleichmäßige Aufteilung der Aufträge sollen die Gesamtarbeitszeiten der Maschinen möglichst gleich sein, d.h. die Differenz zwischen der längsten Gesamtarbeitszeit einer Maschine und der kürzesten Gesamtarbeitszeit einer Maschine soll minimal sein.

Beispiel: Für Aufträge mit den Größen 3,10,6,4 ergibt sich für drei Maschinen die optimale Aufteilung M1: 3, 4; M2: 10; M3: 6.

Schreiben Sie die Klasse `Scheduler` mit der Methode

```
int[] weiseZu(int[] auftraege, int anzMaschinen),
```

die die Zuweisung der Aufträge zu Maschinen zurückliefert. Dabei gibt das Array `auftraege` die Größe der zu verteilenden Aufträge an. Das zurückgelieferte Array enthält für jenen Auftrag die Nummer der Maschine (`1,...,anzMaschinen`), auf der er ausgeführt wird, für obiges Beispiel ist das `{1,2,3,1}`.

Aufgabe 3. Schreiben Sie eine Klasse `Summe` mit einer Methode

```
boolean istSumme(int sum, int[] a),
```

die berechnet, ob es durch Addition von maximal 3 Elementen aus dem Array a möglich ist, genau den Wert sum zu erzielen. Dabei darf ein Element von a nicht mehrfach verwendet werden.

Beispiele: `istSumme(5,{1,3,7})==false`, `istSumme(4,{1,3,7})==true`, `istSumme(21,{13,5,7,3})==true`, `istSumme(28,{13,5,7,3})==false`.