Name:

Matrikelnummer:

Bearbeitungszeit: 120 min.

Aufgabe 1

Gegeben ist die Klasse `Artikel` mit der Methode

```
public boolean istKleinerAls(Artikel a),
```

wobei `a1.istKleinerAls(a2)` genau dann `true` zurück gibt, wenn der Artikel `a1` kleiner als der Artikel `a2` ist. (Das Sortierkriterium ist nicht angegeben.)

Weiters ist die Klasse `Artikelliste` gegeben mit einem parameterlosen Konstruktor, der eine leere Liste anlegt, und den Methoden

```
public void add(Artikel a),
public Artikel get(int i),
public void vertausche(int i, int j),
public int size(),
public String toString().
```

Dabei fügt `add(a)` den Artikel `a` am Ende der Liste hinzu, `get(i)` liefert den Artikel am Listenplatz `i` zurück (die Plätze sind mit 0 beginnend nummeriert), `vertausche(i,j)` vertauscht die Artikel an den Listenplätzen `i` und `j`, `size()` liefert die Größe der Liste, und `toString()` liefert eine Darstellung der Liste.

Schreiben Sie eine Unterklasse `SortierteArtikelliste` von `Artikelliste`, die Artikel mittels `add(a)` so in die Liste einfügt, dass `get(i)` den `(i+1)`-kleinsten Artikel zurück liefert. Weiters soll die Methode `vertausche(i,j)` für eine `SortierteArtikelliste` nichts bewirken. Die Methode `toString()` von `Artikelliste` soll nicht überschrieben werden.

Aufgabe 2

Aufträge vorgegebener Länge sollen auf zwei Maschinen bearbeitet werden, wobei jeder Auftrag nur auf einer der beiden Maschinen bearbeitet werden kann. Weiters muss für die Bearbeitung mancher Aufträge zuvor ein anderer Auftrag fertiggestellt sein.

Beispiel:

Auftragsnummer/Index	0	1	2	3	4	5	6
Bearbeitungszeit	1	1	1	9	10	10	10
Maschine	0	1	1	1	0	0	1
Vorgänger	-1	-1	4	-1	-1	1	0

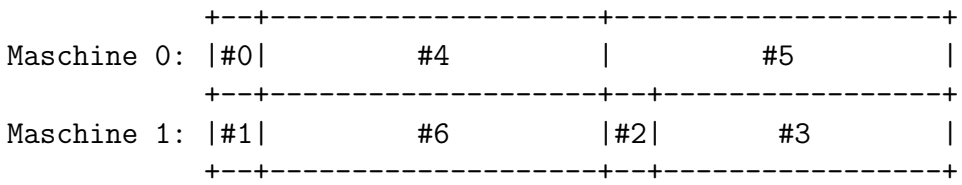
In diesem Beispiel sind 7 Aufträge mit ihren Bearbeitungszeiten gegeben. Die Aufträge 0,4,5 müssen auf Maschine 0 bearbeitet werden, die übrigen Aufträge auf Maschine 1. Auftrag 2 kann erst nach Fertigstellung von Auftrag 4 bearbeitet werden, Auftrage 5 erst nach Fertigstellung von Auftrag 1, und Auftrag 6 erst nach Fertigstellung von Auftrag 0. Ein Vorgänger -1 zeigt an, dass nicht auf die Fertigstellung eines anderen Auftrags gewartet werden muss.

Schreiben Sie eine Klasse `Scheduling` mit der Methode

```
public int[] schedule(int[] bearbeitungszeit,
                      int[] maschine,
                      int[] vorgaenger),
```

die eine optimale Bearbeitungsreihenfolge der Aufträge berechnet, sodass die Fertigstellungszeit des letzten Auftrags minimiert wird. (Die Daten für Auftrag `i` finden sich in `bearbeitungszeit[i]`, `maschine[i]`, und `vorgaenger[i]`.) Dabei wird der nächste Auftrag der Reihenfolge ausgeführt, sobald die entsprechende Maschine frei ist und ein evt. Vorgänger-Auftrag fertiggestellt ist.

Für obiges Beispiel sind z.B. 0,1,4,6,2,5,3 und 0,4,1,6,2,3,5 zwei optimale Reihenfolgen, die beide der folgenden Maschinenbelegung (in zeitlicher Reihenfolge) entsprechen.



Aufgabe 3

Ein Produkt kann von mehreren Lieferanten geliefert werden, wobei die Preise der Lieferanten unterschiedlich sind und jeder Lieferant maximal eine bestimmte Anzahl des Produkts liefern kann. Weiters ist pro Lieferant eine Mindestbestellmenge einzuhalten.

Gegeben ist die Klasse `Lieferant` mit den Methoden

```
public int preis(),
public int lieferbareMenge(),
```

die den Preis des Lieferanten für das Produkt und die vom Lieferanten lieferbare Menge zurück geben.

Schreiben Sie eine Klasse `Bestellung` mit der Methode

```
public int[] bestellen(Lieferant[] lieferanten,
                       int gesamtmenge, int mindestmenge),
```

die die zu bestellende `gesamtmenge` so auf die einzelnen Lieferanten aufteilt, dass der Gesamtpreis minimal ist. Dabei ist die Liste `lieferanten` bereits aufsteigend nach den Preisen der Lieferanten sortiert. Weiters kann jeder Lieferant mindestens die zweifache Mindestbestellmenge liefern. Für `int[] b = bestellen(lieferanten,gesamtmenge,mindestmenge)` soll `b[i]` die Menge angeben, die von `lieferanten[i]` bestellt wird. Dabei zeigt `b[i]==0` an, dass von `lieferanten[i]` nicht bestellt wird. Ansonsten darf `b[i]` die `midestmenge` nicht unterschreiten und die lieferbare Menge des `lieferanten[i]` nicht überschreiten.

Beispiel: Bei Lieferanten

Lieferant	0	1	2	3	4
Preis	11	13	17	19	23
Verfügbare Menge	250	475	225	1000	2000

ergibt sich für die Gesamtmenge 1000 und die Mindestmenge 100 als optimale Bestellung

Lieferant	0	1	2	3	4
Bestellung	250	475	175	100	0

.