Hochschule RheinMain
Prof. Dr. Steffen Reith

Abgabe 2

In dieser Abgabe beschäftigen wir uns mit Backtracking und üben nochmal Rekursion. Bearbeiten Sie dazu die folgenden Teilaufgaben bis zum **Mittwoch 29.6.2022 um 23:59 CEST**. Senden Sie Ihre Lösungen an den jeweiligen Übungsgruppenleiter. Teams sind ausnahmslos nur innerhalb einer Übungsgruppe zulässig!

Aufgabe 1 (Rucksack)

In dieser Aufgabe wollen wir einen Frachtcontainer optimal beladen. Dazu ist eine Liste von Objekten und ihr Gewicht gegeben. In dem nachfolgenden konkreten Beispiel haben wir 4 Objekte:

```
const int NOBJS     = 4;
int gewichte[NOBJS] = {10, 5, 7, 11};
int werte[NOBJS]    = { 7, 6, 2,  1};
}
```

Dabei hat z.B. Objekt 1 in diesem Beispiel das Gewicht 10 und den Wert 7. Ziel der Aufgabe ist es ein Programm zu entwickeln, das die optimale Beladung eines Containers ermittelt. Dabei sollen möglichst große Werte für ein gegebenes maximales Gewicht transportiert werden. Hat der Container also eine maximale Zuladung (Gewicht) von 30, dann kann man den transportierten Wert maximieren indem man die Objekte 1-3 in den Container packt:

```
Der maximale Wert betraegt >15< Einheiten
0: gewaehlt[0] =   1, gewichte[0] =  10, wert[0] =   7
1: gewaehlt[1] =   1, gewichte[1] =   5, wert[1] =   6
2: gewaehlt[2] =   1, gewichte[2] =   7, wert[2] =   2
3: gewaehlt[3] =   0, gewichte[3] =  11, wert[3] =   1
Die benutzte Kapazitaet betraegt >22< Einheiten
```

Entwickeln Sie eine Methode / Funktion

```
int rucksack(int ausgewaehlt[NOBJS],
             int gewichte[NOBJS],
             int werte[NOBJS],
             int restKapa,
             int objIndex)
```

Dabei ist `restKapa` das noch freie Gewicht und `objIndex` der Index des nächstens Objekts von dem eine Zuladung geplant werden soll, d.h. Ihre Suche nach der maximalen Zuladung beginnt immer mit Gesamtkapazität und dem Index

0 (keine Objekte ausgewählt oder abgelehnt). Implementieren Sie `rucksack` mit Hilfe der Methode des **Backtrackings**.

Entwickeln Sie weiterhin eine Methode `greedyRucksack` die eine gierige Strategie zum Beladen des Containers verwendet und vergleichen Sie die Ergebnisse für einige beispielhafte Eingaben.

Hier noch ein Beispiel für eine Containerkapazität von 30:

```
Der maximale Wert betraegt >159< Einheiten
0: gewaehlt[0] =   0, gewichte[0] =  10, wert[0] =   7
1: gewaehlt[1] =   0, gewichte[1] =   5, wert[1] =   6
2: gewaehlt[2] =   0, gewichte[2] =   7, wert[2] =   1
3: gewaehlt[3] =   0, gewichte[3] =  11, wert[3] =   1
4: gewaehlt[4] =   0, gewichte[4] =  13, wert[4] =   1
5: gewaehlt[5] =   1, gewichte[5] =   1, wert[5] =   4
6: gewaehlt[6] =   0, gewichte[6] =   7, wert[6] =  11
7: gewaehlt[7] =   1, gewichte[7] =  11, wert[7] =  20
8: gewaehlt[8] =   0, gewichte[8] =  13, wert[8] =   3
9: gewaehlt[9] =   0, gewichte[9] =  19, wert[9] =   7
10: gewaehlt[10] =   0, gewichte[10] =  19, wert[10] =   8
11: gewaehlt[11] =   0, gewichte[11] =   9, wert[11] =   9
12: gewaehlt[12] =   1, gewichte[12] =   8, wert[12] =  16
13: gewaehlt[13] =   1, gewichte[13] =   2, wert[13] =  19
14: gewaehlt[14] =   1, gewichte[14] =   7, wert[14] = 100
15: gewaehlt[15] =   0, gewichte[15] =  31, wert[15] =   3
Die benutzte Kapazitaet betraegt >29< Einheiten
```

Aufgabe 2 (Rekurrenzen und Laufzeiten)

Bestimmen Sie eine Rekurrenz, die die Worst-Case Laufzeit von `rucksack` beschreibt. Lösen Sie die Rekurrenz, beweisen Sie Ihr Ergebnis mit einer vollständigen Induktion und vergleichen Sie die Laufzeit Ihrer Implementation mit der Abschätzung (überlegen Sie sich welche Containerkapazitäten zum schlechtesten Fall führen!). Bestimmen Sie die Konstante c die durch Ihr Programm festgelegt ist. Führen Sie dazu sinnvolle Zeitmessung mit verschiedenen Anzahlen von Objekten durch.

Aufgabe 3 (Rekursion)

Implementieren und testen Sie eine *rekursive* Methode / Funktion `reverse`, die einen String umdreht, d.h. das Ergebnis von `reverse(Hallo)` ist `ollaH`. Testen Sie Ihre Implementierung auch mit `leohortetrohoel` (mit einer gewissen aktuellen Nebenbedeutung). Stellen Sie eine Rekurrenz für `reverse` auf und lösen Sie diese.

Durchführung:

1. Bilden Sie maximal 2er Teams, wobei 3 oder mehr Personen in einem Team *nicht* zulässig sind. Teams können nur innerhalb einer Gruppe gebildet werden!

2. Lösen Sie die Aufgaben 1-3.

3. Schreiben Sie ein README-File, dass erklärt wie Ihr Projekt, ohne Hilfe einer IDE zu bauen ist. Kann Ihr abgegebenes Projekt nicht gebaut werden oder fehlen entsprechende Hinweise, so kann dies zu einer nicht ausreichenden Wertung führen! Packen Sie den Quellcode, das README und die Ergebnisse der Laufzeittests in ein .zip oder .tgz File.

4. Geben Sie Ihre Lösung (.zip/.tgz) diesmal via EMail bei Ihrem Gruppenleiter ab. Die Betreffzeile Ihrer EMail *muss* die folgende Form haben `Abgabe 2 ADS (Gruppe X): Nachname1_Vorname1, Nachname2_Vorname2`, wobei X Ihre Übungsgruppe bezeichnet. EMails, die man nicht zuordnen kann, werden nicht gewertet.

   Hinweis: Beachten Sie die Bemerkungen zum Thema *Plagiate*. Im Plagiatsfall bekommen mindestens *beide* beteiligte Parteien keine Punkte. Es könnten auch härtere Strafen in Betracht kommen.