HF-ICT - Höhere Fachschule für Informations- und Kommunikationstechnologie
CH 4132 Muttenz
P.Sütterlin

Algorithmen und Datenstrukturen

Exercise sheet 7

---

1 Knacknuss (Kombinatorik)

Lösen Sie folgendes mathematisches Problem: **MATHE + IST + MEIN = LEBEN**

Jeder Buchstabe in der Gleichung representiert eine der gelisteten Variablen:
{A, B, E, H, I, L, M, N, S, T}
Den einzelnen Variabeln sollen folgende Ganzzahlen zugewiesen werden:
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Die Reihenfolge der Zuweisung ist beliebig. Jede Ziffer muss in der Kombination genau einmal vorkommen.

Beispiel: A=9, B=8, E=1, H=2, I=3, L=4, M=5, N=6, S=7, T=0

Setzt man diese Kombination nun in die Gleichung ein, so ergibt dies:
⇒ **59021 + 370 + 5136 = 41816**
⇒ **64527 = 41816**

Schreiben Sie ein Programm welches sämtliche möglichen korrekten Lösungen zu diesem Problem auf den Bildschirm ausgibt:

```
1 class CalcUtil {
2 public:
3    static void printSolutions();
4 };
```

2 Rucksackproblem

Gegeben sind mehrere Lebensmittel mit einem bestimmten Nährwert.

Es soll nun eine Auswahl an Lebensmitteln getroffen werden, die in einen Rucksack mit einer vorgegebenen Gewichtsschranke eingepackt werden können.

Der Nährwert sämtlicher eingepackter Lebensmittel soll maximiert werden so dass der Rucksackträger sich möglichst lange von den Lebensmitteln ernähren kann.

Beispiel:

- 12 kg Sellerie ⇒ Nährwert = 4
- 4 kg Bonbons ⇒ Nährwert = 10
- 2 kg Knäckebrot ⇒ Nährwert = 2
- 1 kg Äpfel ⇒ Nährwert = 2

- 1 kg Karotten ⇒ Nährwert = 1

Bei einer zulässigen Maximallast von 15kg im Rucksack müssten Sie einpacken: **Bonbons, Knäckebrot, Äpfel, Karotten** (Gesamtnährwert = 15 / Tragelast = 8kg).

Schreiben Sie ein Programm welches einen Vektor mit der optimalen Packliste der Lebensmittel zurückgibt:

```
class FoodItem
{
    private:
    string name;
    int weight;
    int nutritionalValue;

    public:
    // Implementieren Sie passende Getter und Setter Methoden
};

class KnapsackUtil
{
    static vector<FoodItem> solveProblem(vector<FoodItem>& foodlist,
                                         int payloadLimit);
};
```

3 Prüfung: Graph ist ein/kein Baum

Schreiben Sie ein Funktion in der Klasse GraphUtil welche ”true” zurückliefert, falls ein ungerichteter Graph ein Baum darstellt - ansonsten geben Sie ”false” zurück:

```
class GraphUtil
{
    static bool checkGraphIsTree(vector<vector<int>>& neighbours);
};
```

Hinweis: Im Gegensatz zum binären Baum darf dieser Baum pro Parent Element null bis *n* Child Elemente besitzen.

4 Maximales Produkt Sub-Array Problem

Suchen Sie in einem Array (vector<int>) jeweils das zusammenhängende Sub-Array welches das grösst mögliche errechenbare Produkt (d.h. die Multiplikation sämtlicher enthaltenen Elementwerte) aufweist. Geben Sie das grösst mögliche Produkt zurück.

Beispiel: {-2, -3, 0, -2, -40}

Das enthaltene Sub-Array mit dem grössten Produkt wäre {-2, -40}.
Der Rückgabewert der Methode wäre somit 80.

Schreiben Sie das passende Programm für diese Aufgabe:

```
class ProblemUtil
{
    static int getMaxSubArrayProduct(vector<int>& values);
};
```