PRG1x & ADE1x Üb. zu Einf. i d. Programmierung und Element. Alg. u. Datenstrukt. **WS 22/23, Übung 4**

Abgabetermin: Sa, 19.11.2022

☐ **Gr. 1,** DI (FH) G. Horn-Völlenkle, MSc **Name** _____ **Aufwand in h** _____

☐ **Gr. 2,** S. Schöberl, MSc

Punkte _____ **Tutor*in / Übungsleiter*in** ____ / ____

1. Funktionen zur Zeichenkettenverarbeitung (4 Punkte)

Implementieren Sie eine Pascal-Funktion `DeleteSubstring`, die alle Vorkommen einer Zeichenkette `substr` aus einer Zeichenkette *s* entfernt und das Ergebnis als Funktionswert liefert. Ist `substr` nicht in `s` enthalten, soll `s` unverändert als Ergebnis geliefert werden.

2. Feldverarbeitung mit offenen Feldparametern: Schnittmengen (8 Punkte)

Gegeben sind zwei beliebig große Felder `a1` und `a2`, die `n1` bzw. `n2` positive ganze Zahlen in aufsteigend sortierter Reihenfolge enthalten. Gesucht ist eine Pascal-Prozedur

```
PROCEDURE Intersect(a1: ARRAY OF INTEGER; n1: INTEGER;
                    a2: ARRAY OF INTEGER; n2: INTEGER;
                    VAR a3: ARRAY OF INTEGER; VAR n3: INTEGER);
```

die das Feld `a3` so belegt, dass darin alle `n3` Zahlen in aufsteigend sortierter Reihenfolge enthalten sind, die sowohl in `a1` als auch in `a2` vorkommen. Im Fehlerfall (Überlauf in `a3`, Felder `a1` und `a2` sind nicht aufsteigend sortiert) soll `n3` auf `-1` gestellt werden.

Beispiel:

a1 =	1	2	3	5	8	13	…	*n1* = 6
a2 =	1	3	5	7	9	…		*n2* = 5
a3 =	1	3	5	…				*n3* = 3

Hinweis: Implementieren Sie eine Pascal-Funktion `IsSorted`, die prüft, ob die Werte in einem Feld `a` aufsteigend sortiert sind und verwenden Sie diese Funktion in der Prozedur `Intersect`.

3. Roulette (5 + 2 + 3 + 2 Punkte)

Schreiben Sie ein Pascal-Programm, das für Sie Roulette spielt. Roulette spielt man, indem man eine Kugel auf ein sich drehendes Rad wirft und wartet, bis die Kugel zufällig in einem der 37 Löcher, die von 0 bis 36 durchnummeriert sind, landet.

a) Entwickeln Sie eine Funktion

```
FUNCTION BenefitForLuckyNr(luckyNr, bet: INTEGER): INTEGER;
```

wobei `luckyNr` die zu setzende Zahl und `bet` den zu setzenden Betrag darstellen. Die Funktion liefert den ausbezahlten Gewinn. Wenn die gesetzte Zahl richtig ist, wird das 36-fache des Einsatzes (`bet`) ausbezahlt, sonst ist der Einsatz verloren.

b) Testen Sie Ihre Funktion wie folgt: Ihr Programm verfügt zu Beginn über 1000 € und soll so lange mit der Funktion `BenefitForLuckyNr` immer wieder 1 € setzen, bis kein Geld mehr übrig ist. Geben Sie nach dem Bankrott Ihres Programms die Anzahl der gewonnenen und verlorenen Spiele sowie den maximalen Geldbetrag, den das Programm während des Spiels jemals "in Händen gehalten" hat, aus. Die Ausgabe könnte dann z. B. wie folgt aussehen:

```
Games lost: 39484
Games won:   1069
Max. money:  1598
```

c) Schreiben Sie eine zweite Funktion

```
FUNCTION BenefitForEvenNr(bet: INTEGER): INTEGER;
```

die das Setzen auf gerade Zahlen simuliert. Im Falle einer geraden Zahl wird der zweifache Einsatz ausbezahlt, im Falle einer ungeraden Zahl oder 0 ist der Einsatz verloren. Beachten Sie, dass beim Roulette die Zahl 0 weder als gerade noch als ungerade angesehen wird.

d) Testen Sie Ihre zweite Funktion `BenefitForEvenNr` so, wie die erste in b). Was fällt Ihnen beim Vergleich der beiden Ergebnisse auf?

Hinweis: verwenden Sie die bereits vorhandenen Pascal-Standardfunktionen für Zufallszahlen:

`Random` gleichverteilte ganzzahlige Zufallszahl im angegebenen Wertebereich oder gleichverteilte Fließkomma-Zufallszahl im Bereich [0,1).
Beispiel für Zufallszahl x mit $1 \leq x \leq 45$:

```
VAR
  x: INTEGER;

x := Random(45) + 1;
```

`Randomize` initialisiert den Zufallszahlengenerator mit der aktuellen Systemzeit

Hinweise:

1. Geben Sie für alle Ihre Lösungen immer eine „Lösungsidee“ an.
2. Dokumentieren und kommentieren Sie Ihre Algorithmen.
3. Bei Programmen: Geben Sie immer auch Testfälle ab, an denen man erkennen kann, dass Ihr Programm funktioniert, und dass es auch in Fehlersituation entsprechend reagiert.