Aufgabe 1: Dank Informatik

Benennen Sie die drei Techniken / Fähigkeiten, die dank Informatik kaum noch benötigt werden. Als Beispiel sei gegeben:

Technikablösung:

Rechenschieber / Fähigkeiten, ihn zu bedienen ersetzt durch Taschenrechner

Technikablösung 1:

Landkarten / Navigation mit Hilfe von Landkarten ersetzt durch Navigationssysteme

Technikablösung 2:

Nachschlagewerke / Nutzung von Nachschlagewerken ersetzt durch Internetsuche

Technikablösung 3:

Füllfederhalter / Erzeugung von handschriftlichen Texten ersetzt durch Textsysteme

Name:_____ Matr.-Nr: _____

Aufgabe 2: Grammatiken

Gegeben sei eine Grammatik wie folgt (ε ist Leersymbol):

G = <Startsymbol, Terminale, Nicht-Terminale, Regeln> mit

Startsymbol = *Start*

Terminale = {*a, b, c, d*}

Nicht-Terminale = {*X, Y* }

Regeln = {*Start* $\rightarrow$ *X,*
Start $\rightarrow$ *Y,*
Start $\rightarrow$ *XY,*
X $\rightarrow$ *ab,*
X $\rightarrow$ *abX,*
Y $\rightarrow$ *cd,*
Y $\rightarrow$ *cYd* }

Entscheiden Sie, ob folgende Wörter in der durch die Grammatik definierten Sprache enthalten sind:

- ε ~~ja~~ / nein
- abba ~~ja~~ / nein
- acdc ~~ja~~ / nein
- abcd ja / ~~nein~~
- aabccd ~~ja~~ / nein
- aabbccdd ~~ja~~ / nein
- ababccdd ja / ~~nein~~
- abab ja / ~~nein~~
- cd ja / ~~nein~~
- ababacd ~~ja~~ / nein

Bestimmen Sie die Wörter, die durch die Grammatik akzeptiert werden in der Art $a^n b^n$:

$[a,b]^+ \cup c^m d^m \cup [a,b]^+ c^n d^n$ mit m,n > 0

Name:_____ Matr.-Nr: _____

Aufgabe 3: ADT Stack

Implementieren Sie einen Stack in Pseudocode auf Basis eines Arrays mit den Funktionen:

- size() gibt die Anzahl der Elemente im Stack zurück
- isEmpty() liefert **true**, wenn der Stack leer ist, **false** sonst
- top() liefert das oberste Element des Stacks, ohne es zu entfernen, **false**, falls der Stack leer ist
- push(Obj) fügt das Element **Obj** oben in den Stack ein, **true**, falls erfolgreich, **false**, falls der Stack voll ist
- pop() liefert das oberste Element des Stacks und entfernt es aus dem Stack, **false**, falls der Stack leer ist.

```
declare max := 100, count := 0, array[1..max].

size() {return count}

isEmpty() {return if (count = 0) true else false}

top() {return if (count = 0) false else array[count]}

push(Obj) {if (count = max) return false
           else {count=count+ 1; array[count] := Obj; return true}}

pop() {if (count=0) return false
       else {count := count-1; return array[count+ 1]}}
```

Name:_____ Matr.-Nr: _____

Aufgabe 4: Prolog Unifikation

Gegeben seien jeweils 2 Prolog-Ausdrücke. Sie sollen entscheiden, ob diese beiden Ausdrücke unifizierbar sind und, falls dieses der Fall ist, den durch Substitution erzeugten Ausdruck angeben. Sollte die Unifikation misslingen, geben Sie den Grund an. Als Beispiele sind gegeben (Konstanten fangen mit kleinem, Variablen mit großem Buchstaben an):

- test (X, foo, Y) = test (M, N, M) ☺ mit test (X, foo, X).
- test (kai, X) = test1(kai, udo) ☹ weil test ≠ test1.

Hier nun die Ausdrücke:

- [Esther, koenig] = [roland, Seiffert].

☺ mit Esther = roland, Seiffert = koenig.

- esther(Koenig) = roland(Seiffert).

☹ esther ≠ roland.

- member(3, [2, 4, 6]) = member(X, [2, 4, 6]).

☺ mit X = 3.

- member(3, L), L = [1, 2, 3, 4].

☺ mit L = [1, 2, 3, 4].

Name:_____ Matr.-Nr: _____

Aufgabe 5: Prolog Programmanalyse

Gegeben ein Prolog-Programm `foo(List1, Elem1, List2, Elem2)` wie folgt

```
foo(List,Elem1,List,Elem2) :-
        not(member(Elem1,List)), not(member(Elem2,List)).
foo(List1,Elem1,List2,Elem2) :-
        append(First,[Elem1|Rest],List1),
        append(First,[Elem2|Rest],List2).
```

Beschreiben Sie kurz, was das Programm macht:

Das Programm `foo` evaluiert zu wahr, wenn `List1` und `List2` unifiziert werden können durch den Austausch von `Elem1` in `List1` mit `Elem2` in `List2`.

Zu was unifiziert `List` in den folgende Beispielen:

```
foo([1, 2, 3, 4], 3, List, kai).
```

List = [1, 2, kai, 4].

```
foo(List, kai, [1, 2, 3, 4], 3).
```

List = [1, 2, kai, 4].

```
foo([1,2,3,4], 5, List, kai).
```

List = [1, 2, 3, 4].

Name:_____ Matr.-Nr: _____

Bewertung der Aufgaben

Aufgabe 1:	irgendwie zutreffende Beschreibung (maximal 3 Punkte)	=	1 Punkt
Aufgabe 2:	pro richtige Wortentscheidung	=	0.5 Punkte
	fast richtiger regulärer Ausdruck	=	0,5 Punkte
	richtige regulärer Ausdruck (maximal 6 Punkte)	=	1 Punkt
Aufgabe 3:	richtiger Code	=	1 Punkt / Code
	fast richtiger Code	=	0,5 Punkt /Code
	richtiger Ansatz	=	2 Punkte
	fast richtiger Ansatz	=	1 Punkt
	(maximal 5 Punkte)		
Aufgabe 4:	jeweils richtiges Ergebnis (maximal 4 Punkte)	=	1 Punkt
Aufgabe 5:	richtige Codebeschreibung	=	2 Punkte
	fast richtige Codeskizze	=	1 Punkte
	richtige Beispiele	=	1 Punkt / Beispiel
	(maximal 5 Punkte)		
Maximale Punkte aller Aufgaben:		=	23 Punkte

Notengebung: Maximum(Minimum(Aufrunden(Summe aller Punkte - 4),15),0)

Name:_____ Matr.-Nr: _____