hochschule hof
University of Applied Sciences

- Vom Prüfungsteilnehmer vollständig auszufüllen -

~~Winter~~ / Sommer – Semester 2021

Studiengang : _____

Prüfungsfach: Formale Sprachen _____

Matrikelnummer: | | | | | | | | |

Platznummer	**Raumnummer/ Turnhalle**	
_____	_____	

Prüfungstag:	
Arbeitszeit:	
Beginn der Bearbeitung:	
Ende der Bearbeitung:	

Vom Prüfer/in bzw. 1. Korrektor/in auszufüllen

(Note) | Unterschrift des Erstprüfer/s/in

Vom 2. Korrektor/in auszufüllen

(Note) | Unterschrift des Zweitprüfer/s/in

Erlaubte Hilfsmittel:

_____

_____

_____

Bemerkungen (z.B. Unterbrechung der Bearbeitung, wird vom Aufsichtführenden ausgefüllt)

von _____ **Uhr bis** _____ **Uhr**

von _____ **Uhr bis** _____ **Uhr**

von _____ **Uhr bis** _____ **Uhr**

_____

_____

_____

1. Aufgabe Maschinenmodelle

	10

1. Auf dem Eingabeband einer RAM steht ein einzelnes Zeichen. Die beiden möglichen Zeichen können ‚A' oder ‚B' sein. Entwickeln Sie ein Programm für die RAM, welches dieses Zeichen 100-mal auf das Ausgabeband schreibt!

6

1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	
11.	
12.	
13.	
14.	
15.	
16.	
17.	
18.	
19.	
20.	
21.	
22.	
23.	
24.	

2. Auf dem Arbeitsband einer Turingmaschine steht eine Dualzahl. Entwickeln Sie ein Programm, das jede Ziffer '0' durch die Ziffer '1' ersetzt!

   Beispiele:

   `10101` $\rightarrow$ `11111`

   `101` $\rightarrow$ `111`

   Für das Programm können Sie das Zeichen `'□'` benutzen, um den Inhalt einer leeren Zelle anzugeben. Benutzen Sie als Startzustand den Zustand 'S'! Der Kopf steht am Anfang auf dem ersten Zeichen der Dualzahl (links).

4

Bedingung		Aktion	
Alter Zustand	Kopf	Neuer Zustand	Aktion

2. Aufgabe Komplexität

	8

1. Sie untersuchen die Zeitkomplexität für ein Entscheidungsproblem A. Sie haben dazu einen Algorithmus entwickelt, der dieses Problem mit kubischen Zeitufwand löst. Was können Sie daraus für die Komplexität dieses Problems folgern?

2

2. Für das Entscheidungsproblem A finden Sie zwei weitere Entscheidungsprobleme B und C, so dass sich die Eingaben für die beiden Entscheidungsprobleme B und C in eine Eingabe für A transformieren und mit einem Algorithmus für A lösen lassen. Das Entscheidungsproblem B hat eine quadratische und das Entscheidungsproblem C eine kubische Zeitkomplexität. Der Zeitaufwand für die Transformation von B nach A wächst exponentiell und für die Transformation von C nach A linear mit Größe der Eingabe. Was können Sie mit diesen beiden Transformationen für die Zeitkomplexität von A folgern?

4

3. Mit welchem Zeitaufwand kann ein endlicher Automat als Akzeptor das Wortproblem einer CH-3-Sprache lösen?

2

3. Aufgabe Grammatiken

	12

1. Geben Sie eine Grammatik für eine Sprache mit den Zeichen '1', '0', '(' und ')' an! Ein Wort dieser Sprache ist eine geklammerte Dualzahl. Das erste Zeichen ist also immer eine öffnende Klammer und das letzte Zeichen eine schließende Klammer. Dazwischen steht eine beliebige Sequenz der Zeichen '1' und '0'. Es muss mindestens eines dieser Zeichen zwischen den Klammern vorkommen. (3)

   Beispiele:

   - (0)
   - (101)
   - (0010)
   - (111111)

2. In der Vorlesung wurde der Beweis für das Pumping-Lemma skizziert. Wozu wurde dieses Lemma genutzt? (3)

3. Analysieren Sie die folgende Grammatik und geben Sie drei Wörter der zugehörigen Sprache an! Beschreiben Sie danach möglichst genau die gesamte Menge der Wörter, die für diese Grammatik erzeugt werden! Die Nichtterminalsymbole sind hier 's', 'x', 'y', 'z', 'p' und 'q' wobei 's' das Startsymbol ist. Die Terminalsymbole sind 'a' und 'b'. (6)

   $s \rightarrow xpyqz$
   $p \rightarrow a$
   $p \rightarrow ap$
   $q \rightarrow b$
   $q \rightarrow bq$
   $xa \rightarrow ax$
   $xy \rightarrow x$
   $xb \rightarrow ax$
   $xbz \rightarrow a$

4. Aufgabe Rekursiv absteigender Parser

	12

Die folgende Grammatik beschreibt eine Zuweisung, bei der einer Variablen der Wert eines booleschen Ausdrucks zugewiesen wird. Der einzige Operator für diesen Ausdruck ist das logische XOR, der mit dem Symbol ‚^' dargestellt wird. In dem Ausdruck dürfen die Konstanten ‚1' und ‚0' für ‚true' und ‚false', Variablen sowie ein geklammerter Ausdruck vorkommen. Diese Ausdrücke haben die folgende Form:

```
<assign>     ::= <var> = <expr>
<expr>       ::= <literal> {^ <literal> }
<literal>    ::= 1 | 0 | <var> | '(' <expr> ')'
<var>        ::= <character> { <character> | <digit> }
```

Ein Beispiel entsprechend dieser Grammatik könnte wie folgt aussehen:

x1=y^(z^1)

Ergänzen Sie den folgenden Code an den markierten Stellen so, dass der Parser eine solche Zuweisung einliest und das Ergebnis berechnet und der Variable zuweist! Dazu sollen insbesondere die Funktionen „parseAssign" und „parseExpr" implementiert werden. Die Funktion „parseExpr" sowie weitere Funktionen geben jeweils das Ergebnis der einzelnen Operationen als booleschen Wert zurück. Die Funktion „parseAssign" weist den berechneten Wert der angegebenen Variable zu und nutzt dazu die HashMap „values".

```
public class LogicParser {
  static HashMap<String,Boolean> values = new HashMap<String,Boolean>();
  int c;
  FileReader fr;

  public LogicParser (String fileName) throws IOException {
    fr = new FileReader(fileName);
    read();
  }

  public void read() throws IOException {
    c = fr.read();
    System.out.print((char) c);
  }

  // <assign> ::= <var> = <expr>
  void parseAssign() throws Exception {
```

```
  }

  // <expr> ::= <literal> {^ <literal> }
  boolean parseExpr() throws Exception {
```

```
  }

  // <literal> ::= 1 | 0 | <var> | '(' <expr> ')'
  boolean parseLiteral() throws Exception {
    if (c == '1') {
      read();
      return true;
    }
    else if (c == '0') {
      read();
      return false;
    }
    else if (Character.isLetter(c)) {
      return values.get(parseVar());
    }
    else {
      if (c != '(') throw new Exception("Literal erwartet: " + c);
      read();
      boolean result = parseExpr();
      if (c != ')') throw new Exception("Zeichen ')' erwartet: " + c);
      read();
      return result;
    }
  }

  // <var> ::= <character> { <character> | <digit> }
  String parseVar() throws Exception {
    StringBuffer sb = new StringBuffer();
    if (! Character.isLetter(c))
      throw new Exception("Buchstabe erwartet: " + c);
    sb.append((char) c);
    read();
    while (Character.isLetter(c) || Character.isDigit(c)) {
      sb.append((char) c);
      read();
    }
    return sb.toString();
  }
}
```

5. Aufgabe XML-Parser

	8

Eine XML-Dateien enthält die Noten für eine Prüfung. Dazu enthält das Element „grades“ ein oder mehrere Elemente mit der Bezeichnung „student“. Der Inhalt dieses Elements „student“ ist der Name des Studenten oder der Studentin. Die Note wird für dieses Element mit dem Attribut „grade“ angegeben.

Ein Beispiel für eine solche Datei könnte wie folgt aussehen:

```
<grades>
  <student grade="1">John Smith</student>
  <student grade="2">Mary Baker</student>
  <student grade="3">Jeff Miller</student>
  <student grade="4">Emma Brown</student>
  <student grade="5">Jerry White</student>
</grades>
```

Entwickeln Sie ein Programm, das für eine solche Datei die Durchschnittsnote berechnet und ausgibt! Für das obige Beispiel wäre das Ergebnis 3.0. Ergänzen Sie dazu den folgenden Programmcode an der markierten Stelle!

```
public class NotenParser {
  Document doc;

  public NotenParser(File file) throws Exception {
    DocumentBuilderFactory
      dbf = DocumentBuilderFactory.newInstance();
    dbf.setValidating(true);
    dbf.setIgnoringComments(true);
    dbf.setIgnoringElementContentWhitespace(true);
    dbf.setExpandEntityReferences(true);
    dbf.setNamespaceAware(true);
    DocumentBuilder db = dbf.newDocumentBuilder();
    db.setErrorHandler(new DefaultHandler());
    doc = db.parse(file);
  }

  public void parseFile() {
```

```
  }

  public static void main(String[] args) {
    NotenParser parser;
    try {
      parser = new NotenParser(new File(args[0]));
      parser.parseFile();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```