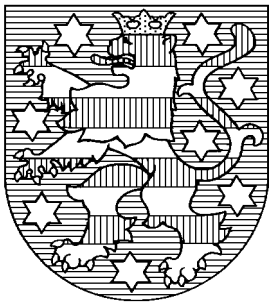
Thüringer Kultusministerium



Abiturprüfung 2000

Leistungsfach

Informatik

(Haupttermin)

Arbeitszeit: 270 Minuten

Hilfsmittel: Formeln und Tabellen für die Sekundarstufen I und II/
Paetec, Gesellschaft für Bildung und Technik mbH, Berlin;
PC mit Prolog-System und
Turbo Pascal- oder Oberon-System;
Taschenrechner

Der Prüfungsteilnehmer löst **alle** Aufgaben.
Rechts unten neben jeder Teilaufgabe steht die für diese Teilaufgabe maximal erreichbare Anzahl von Bewertungseinheiten (BE).
Der Prüfungsteilnehmer sichert bei der praktischen Arbeit am PC mindestens alle 10 Minuten die von ihm erarbeiteten Programme.
Der Prüfungsteilnehmer hat die von ihm erarbeiteten Programme und Module im Quelltext zu kommentieren.
Der Prüfungsteilnehmer hat die von ihm erarbeiteten Quelltexte abzugeben.

Öffnung am 05. Mai 2000

1 E-Mail-Adressen

E-Mail-Adressen sind nach syntaktischen Regeln aufgebaut. Die folgenden drei Beispiele geben die typische Struktur von E-Mail-Adressen wieder:

tkm@thueringen.de
thomas.mustermann@gym.ef.th.schule.de
dia@gi-ev.de

Eine E-Mail-Adresse soll nur aus Kleinbuchstaben und aus den folgenden Zeichen aufgebaut sein:

@	.	-

1.1 Geben Sie einen endlichen Automaten an, der die folgende Sprache akzeptiert!
Die Menge aller Zeichenfolgen, die eine korrekte E-Mail-Adresse darstellen. Dabei wird davon ausgegangen, dass eine Zeichenfolge stets durch das Zeichen „~“ abgeschlossen ist. 4 BE

1.2 Erläutern Sie an Hand der folgenden E-Mail-Adresse, wie der von Ihnen in Teilaufgabe 1.1 angegebene endliche Automat arbeitet:
hans.mueller@institution.gth.shuttle.de 2 BE

1.3 Beschreiben Sie die Syntax von E-Mail-Adressen!
Geben Sie dazu eine Grammatik an! 4 BE

1.4 Erläutern Sie an Ihrer Syntaxbeschreibung von Teilaufgabe 1.3 die Begriffe Terminalsymbol, Nichtterminalsymbol, Startsymbol und Produktionsregel! 4 BE

1.5 Entscheiden Sie, ob es sich bei der von Ihnen in Teilaufgabe 1.3 angegebenen Grammatik um die Grammatik einer regulären Sprache oder um die Grammatik einer kontextfreien Sprache handelt!
Begründen Sie Ihre Entscheidung! 3 BE

2 Das one-time Pad

2.1 Ein Klartext ist eine Folge von Klarbuchstaben.
Klarbuchstaben sind Kleinbuchstaben.
Ein Geheimtext ist eine Folge von Geheimbuchstaben.
Geheimbuchstaben sind Großbuchstaben.
Ein Schlüssel ist eine Folge von Schlüsselbuchstaben.
Schlüsselbuchstaben sind Kleinbuchstaben.

Die Verschlüsselung eines Klarbuchstabens wird an dem folgenden Schema erläutert:

Klarbuchstabe	Schlüsselbuchstabe: abcdefghij**k**lmnopqrstuvwxyz
a	ABCDEFGHIJKLMNOPQRSTUVWXYZ
b	BCDEFGHIJKLMNOPQRSTUVWXYZA
c	CDEFGHIJKLMNOPQRSTUVWXYZAB
d	DEFGHIJKLMNOPQRSTUVWXYZABC
e	EFGHIJKLMNOPQRSTUVWXYZABCD
f	FGHIJKLMNO**P**QRSTUVWXYZABCDE
g	GHIJKLMNOPQRSTUVWXYZABCDEF
h	HIJKLMNOPQRSTUVWXYZABCDEFG
...	
x	XYZABCDEFGHIJKLMNOPQRSTUVW
y	YZABCDEFGHIJKLMNOPQRSTUVWX
z	ZABCDEFGHIJKLMNOPQRSTUVWXY

Abbildung 1

Der Klarbuchstabe bestimmt die Zeile und der Schlüsselbuchstabe bestimmt die Spalte, in der sich der Geheimbuchstabe befindet.

Beispiel:
Der Klarbuchstabe `f` wird mit dem Schlüsselbuchstaben `k` verschlüsselt. Der Geheimbuchstabe `P` steht in der Zeile `f` und der Spalte `k`. Die drei Buchstaben sind in der Abbildung 1 fett hervorgehoben.

a) Entwerfen Sie ein Modul krypto , das die folgenden Operationen realisiert:

- Verschlüsseln eines Klarbuchstabens,
- Entschlüsseln eines Geheimbuchstabens!

Implementieren Sie das Modul in **Turbo Pascal** oder **Oberon**!

Beachten Sie die folgenden Festlegungen:

- Beim Verschlüsseln wird für einen Klarbuchstaben und einen Schlüsselbuchstaben der dazugehörige Geheimbuchstabe ermittelt.

- Beim Entschlüsseln wird für einen Geheimbuchstaben und einen Schlüsselbuchstaben der dazugehörige Klarbuchstabe ermittelt.

- Das Ver- und Entschlüsseln erfolgen nach den Festlegungen der Abbildung 1.

6 BE

b) Entwerfen Sie ein Programm demo , in dem die beiden Operationen des Moduls krypto getestet werden!
Implementieren Sie das Programm in **Turbo Pascal** oder **Oberon**!
Das Programm demo hat das Modul krypto zu importieren.

4 BE

2.2 Entwerfen Sie ein Programm key ! Das Programm soll mit Hilfe eines Zufallszahlengenerators einen Schlüssel erzeugen, der aus 100 Schlüsselbuchstaben besteht, und diesen in einer Turbo Pascal-Textdatei oder in einem Oberon-Textfenster abspeichern.
Implementieren Sie das Programm in **Turbo Pascal** oder **Oberon**!

8 BE

2.3 Es gibt ein sehr sicheres Verschlüsselungsverfahren, das die folgenden Eigenschaften besitzt: Klartext, Schlüssel und Geheimtext besitzen die gleiche Länge und ein Schlüssel wird nur einmal verwendet.

Beispiel:
Klartext: `esistallesvollkommeninordnung`
Schlüssel: `tgkelffpoxslkdeplawedfjfngidf`
Geheimtext: `XYSWEFQASPNZVOODXMARLSXWQTCQL`

Klartext, Schlüssel und Geheimtext besitzen jeweils die Länge 29. Das Verschlüsseln der Klarbuchstaben und das Entschlüsseln der Geheimbuchstaben erfolgen so, wie es mit der Abbildung 1 beschrieben wurde.

Entwerfen Sie ein Programm otp , das die folgenden Operationen realisiert:
- Prüfen, ob ein Klartext korrekt ist,
- Verschlüsseln eines Klartextes,
- Prüfen, ob ein Geheimtext korrekt ist,
- Entschlüsseln eines Geheimtextes.

Implementieren Sie das Programm in **Turbo Pascal** oder **Oberon**!
Das Programm otp hat das Modul krypto zu importieren.

Beachten Sie die folgenden Festlegungen:

- Beim Verschlüsseln wird der Klartext mit Hilfe des in Teilaufgabe 2.2 erzeugten Schlüssels in einen Geheimtext überführt.

- Beim Entschlüsseln wird der Geheimtext mit Hilfe des in Teilaufgabe 2.2 erzeugten Schlüssels in einen Klartext überführt.

- Ein Klartext bzw. ein Geheimtext sind korrekt, wenn sie ausschließlich aus der jeweiligen Zeichenart bestehen und höchstens 100 Zeichen besitzen.

- Klartext und Geheimtext sind in Turbo Pascal-Textdateien oder in Oberon-Textfenstern abzuspeichern.

- Die vier Operationen des Programms otp sind zu testen!

10 BE

3 Auskunftssystem

Die Klassikerstraße Thüringens führt durch die Städte Weimar, Arnstadt, Ilmenau, Meiningen, Eisenach, Gotha, Erfurt, Jena und Rudolstadt. Abbildung 2 stellt den Verlauf der Klassikerstraße mit den Entfernungen zwischen benachbarten Städten dar.

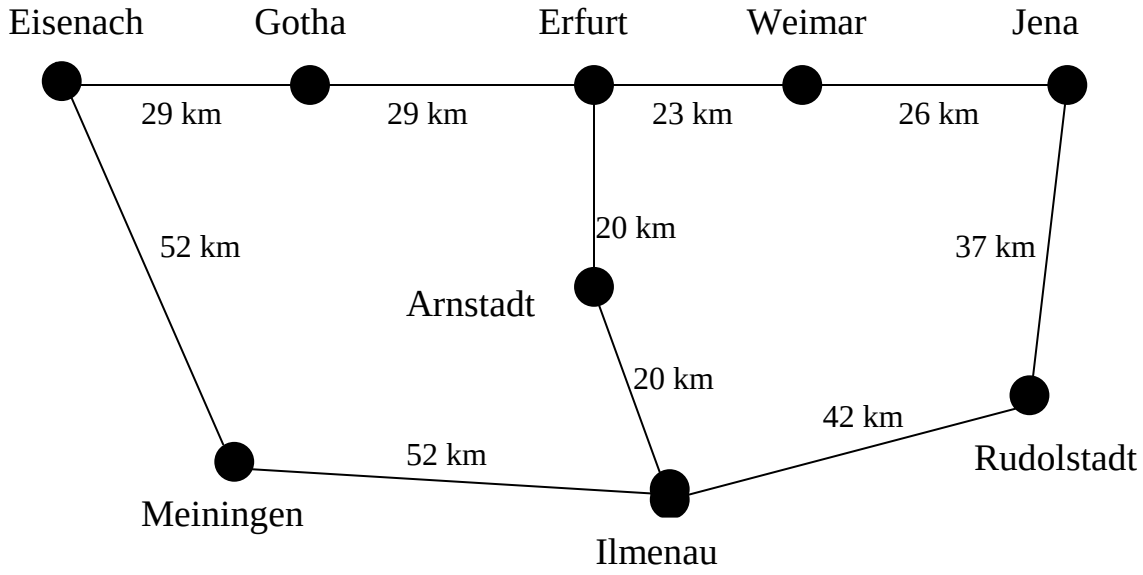


Abbildung 2

Entwerfen und implementieren Sie ein **Prolog-Programm**, das bei Abfrage alle Fahrtrouten von einer Startstadt zu einer Zielstadt auf der Klassikerstraße sucht!

Beachten Sie die folgenden Festlegungen:

- Jede Stadt, durch die die Klassikerstraße führt, kann Startstadt sein.
- Zielstadt muss eine Stadt sein, durch die die Klassikerstraße führt und die nicht mit der Startstadt identisch ist.
- Eine Stadt darf höchstens einmal durchfahren werden.
- Vom Programm sind für jede Fahrtroute auszugeben:
  - die Städte, die auf der Fahrtroute liegen (in der Reihenfolge von der Startstadt bis zur Zielstadt) und
  - die Entfernung von der Start- bis zur Zielstadt.

15 BE