IN-G 2014 HT

Saarland
Ministerium
für Bildung und Kultur

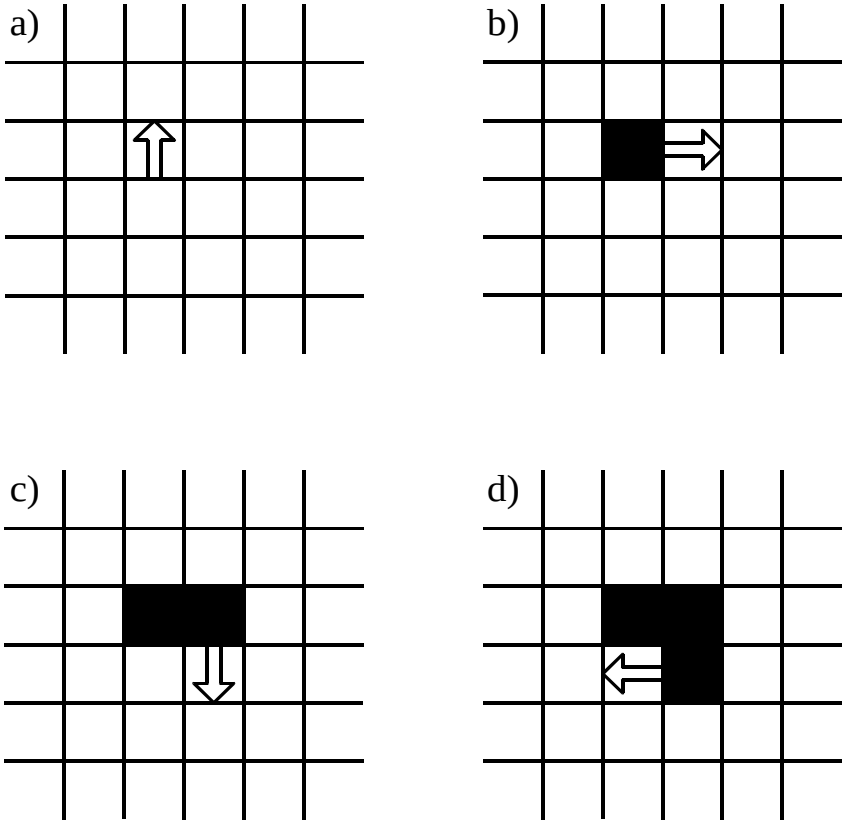
Schriftliche Abiturprüfung 2014

Fach:	**Informatik Haupttermin**
Kurstyp:	G-Kurs
Datum:	12.05.2014
Bearbeitungszeit:	3 Zeitstunden
Hilfsmittel:	nicht-programmierbarer Taschenrechner
Seitenzahl:	Die Prüfungsaufgabe umfasst mit Deckblatt und Anlagen 8 Seiten.

Hohenzollernstraße 60	66117 Saarbrücken
www.saarland.de	

1. Aufgabe

Im Jahr 1986 erfand der Amerikaner Chris Langton einen theoretischen *Automaten*, der aus einem unendlich großen *Spielfeld* besteht, das in quadratische *Zellen* unterteilt ist, die zunächst alle weiß gefärbt sind. Auf einer der Zellen befindet sich eine sogenannte *Ameise*, die hier als Pfeil dargestellt ist. Die Pfeilrichtung gibt die Blickrichtung der Ameise an.



Jeder *Ausführungsschritt* verläuft nun wie folgt:

i. Ist die Zelle weiß, auf der sich die Ameise befindet, so dreht sich die Ameise um 90° nach rechts, ansonsten um 90° nach links.

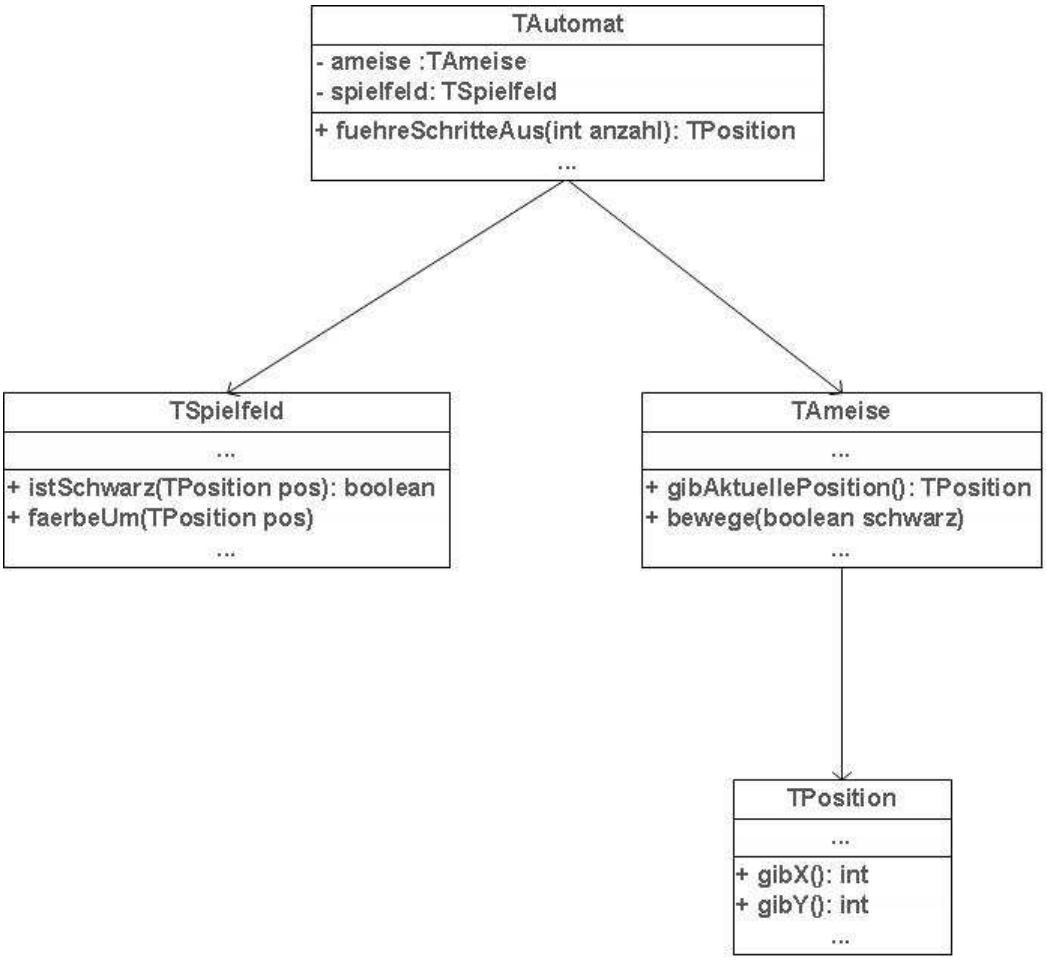
ii. Die entsprechende Zelle wird umgefärbt: Weiß wird zu schwarz, schwarz zu weiß.

iii. Die Ameise bewegt sich um eine Zelle nach vorne.

Die Abbildungen a) bis d) zeigen exemplarisch die Ausgangssituation sowie die ersten drei Ausführungsschritte.

1.1 Zeichnen Sie vier weitere Ausführungsschritte des in der Abbildung dargestellten Automaten.

Das unten abgebildete Diagramm modelliert Langtons Automaten objektorientiert.
(Hinweis: Die Dokumentation der beteiligten Klassen befindet sich in Anhang 1.)



1.2 Objekte der Klasse `TPosition` beinhalten eine x- und eine y-Koordinate und repräsentieren somit die Position einer Zelle auf dem Spielfeld.
Implementieren Sie die Klasse `TPosition` mit Konstruktor und den Methoden `gibX` und `gibY`.

1.3 Implementieren Sie die Methode `fuehreSchritteAus` der Klasse `TAutomat`.

1.4 Implementieren Sie die Methoden `istSchwarz` und `faerbeUm` der Klasse `TSpielfeld` für den Fall, dass das Spielfeld intern mit Hilfe eines zweidimensionalen Feldes namens `feld` verwaltet wird.

1.5 Die Verwaltung des unendlich großen Spielfeldes mit Hilfe eines zweidimensionalen Feldes scheint zwar nahe zu liegen, bringt aber ein konkretes Problem mit sich. Beschreiben Sie das auftretende Problem und entwickeln Sie eine alternative Möglichkeit, das Spielfeld in der Klasse `TSpielfeld` zu verwalten, durch die das Problem umgangen werden kann.

2. Aufgabe

2.1 Binäre Bäume

2.1.1 Ein geordneter, binärer Baum wird auch als „binärer Suchbaum“ bezeichnet. Geben Sie allgemein die Eigenschaften eines solchen binären Suchbaumes an, wenn jeder seiner Knoten eine natürliche Zahl enthält.

2.1.2 Zeichnen Sie denjenigen binären Suchbaum, der entsteht, wenn die Zahlen

5, 4, 2, 8, 9, 6, 1, 7, 3

der Reihe nach in den zunächst leeren binären Suchbaum eingefügt werden.

2.1.3 Geben Sie jeweils die Sequenz der Knoten des binären Suchbaums aus 2.1.2 an, wenn er in preorder- bzw. in inorder-Reihenfolge durchlaufen wird.

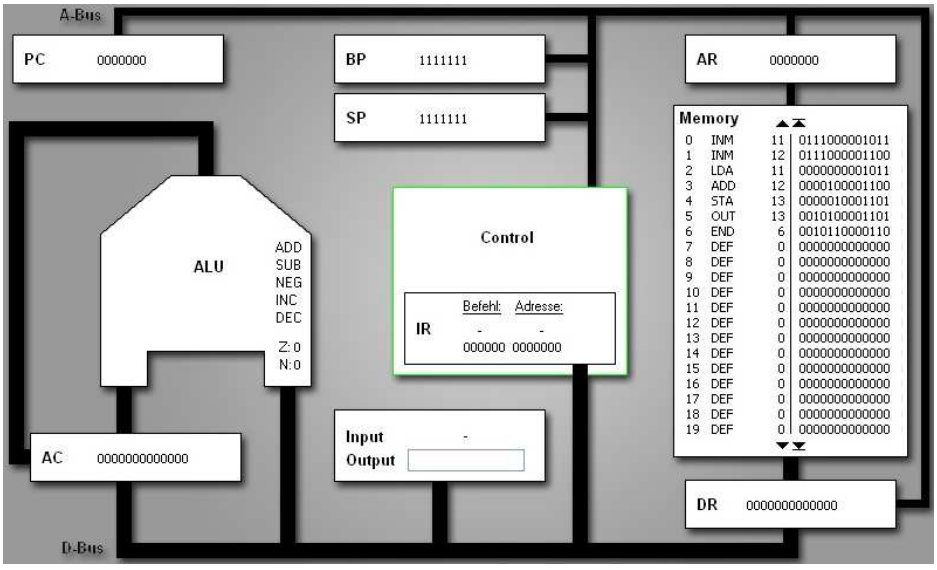
2.1.4 Entwickeln Sie graphisch denjenigen binären Baum, dessen preorder- bzw. inorder-Durchläufe folgende Sequenzen liefern:

8, 2, 6, 5, 1, 3 bzw. 6, 2, 5, 8, 1, 3

2.1.5 Begründen Sie, weshalb beim inorder-Durchlauf eines binären Suchbaums stets eine sortierte Sequenz entsteht.

2.2 Simulationsprogramm DC

Das Programm DC simuliert auf dem Bildschirm eine einfache Zentraleinheit (CPU) nach dem von-Neumann-Prinzip. Folgende Abbildung zeigt einen Screenshot des DC: (Hinweis: Der Befehlssatz des DC befindet sich in Anhang 2.)



2.2.1 Beschreiben Sie alle Phasen des Instruktionszyklus bei der Ausführung eines Befehls durch den Prozessor.

2.2.2 Implementieren und kommentieren Sie ein DC-Programm, das zwei positive ganze Zahlen *a* und *b* einliest und das ganzzahlige Ergebnis der Division von a und b sowie den zugehörigen Rest der Division ausgibt.

3. Aufgabe

Im Zeitalter des Internets und gerade in sozialen Netzwerken spielt die Geheimhaltung der privaten Informationen und Nachrichten eine immer größere Rolle.

3.1 Grammatiken

Ein erster Schutz zur Abwehr Dritter in sozialen Netzwerken ist ein gut gewähltes Passwort für den Login. Relativ sichere Passwörter bestehen aus Klein- und Großbuchstaben, Ziffern, Umlauten und Sonderzeichen. Um eine möglichst hohe Anzahl möglicher Kombinationen zu erhalten, sollte ein Passwort eine bestimmte Mindestlänge besitzen.

Passwörter können mit einer Grammatik dargestellt werden. Es gelten die folgenden vereinfachten Regeln:

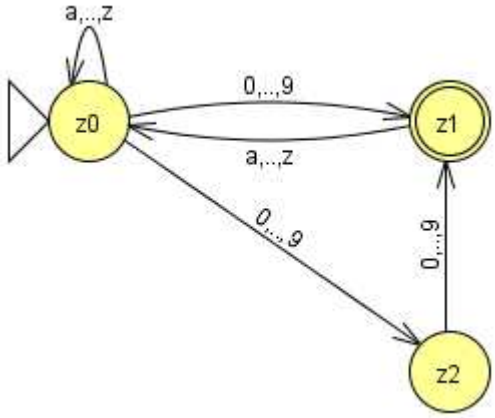
- Länge des Passwortes: genau 4 Zeichen
- Zeichen:
  - Sonderzeichen @,!,$,?,&,%
  - Ziffern 0,..,9
  - Kleinbuchstaben a,..,z
- Keine Sonderzeichen und keine Ziffern als erstes und letztes Zeichen.
- Maximal zwei Kleinbuchstaben nacheinander.

3.1.1 Gegeben sind vier Passwörter. Überprüfen Sie diese mit Hilfe der obigen Regeln auf Korrektheit und begründen Sie bei Unkorrektheit, welche Regel verletzt wurde.
a) m@$a b) gak4 c) r#7s d) i?fo

3.1.2 Entwickeln Sie zu obigen Regeln eine reguläre Grammatik $G_{PW-Generator}$.

3.2 Automaten

Gegeben ist der folgende nicht-deterministische endliche Automat ($NEA_{Passwortcheck}$), der die Funktion eines "Passwortcheckers" für Passwörter übernimmt.



3.2.1 Erläutern Sie, welche Art von Passwörtern vom $NEA_{Passwortcheck}$ akzeptiert werden.

3.2.2 Entwickeln Sie mit Hilfe der Teilmengenkonstruktion den Übergangsgraphen eines deterministischen endlichen Automaten $DEA_{Passwortcheck}$, der die gleiche Sprache akzeptiert wie der angegebene nicht-deterministische endliche Automat $NEA_{Passwortcheck}$.

3.3 Kryptographie - Playfairchiffre

Eine weitere Möglichkeit die privaten Nachrichten zu schützen, ist die Anwendung eines kryptographischen Verfahrens auf die unverschlüsselte Nachricht. Dadurch wird die verschlüsselte Nachricht "unlesbar" für Dritte. Ein historisches Beispiel ist das *Playfair*-Verfahren:
Exemplarisch wird der Schlüssel „*PLAYFAIRCHIFFRE*“ betrachtet. Doppelte Buchstaben des Schlüssels werden entfernt und die verbleibenden Buchstaben des Schlüssels zeilenweise, von links nach rechts, in eine Schlüsseltabelle eingetragen. Der Rest der Schlüsseltabelle wird durch die noch nicht verwendeten Buchstaben des Alphabets zeilenweise ergänzt:

P	L	A	Y	F
I/J	R	C	H	E
B	D	G	K	M
N	O	Q	S	T
U	V	W	X	Z

Die zu verschlüsselnde Botschaft wird nun in Buchstabenpaare zerlegt. Lautet die Botschaft zum Beispiel „*GEHEIMTEXT*“, so wird sie zerlegt in: „*GE HE IM TE XT*“. Werden nun die Buchstaben der jeweiligen Buchstabenpaare in der Schlüsseltabelle gesucht, so ergeben sich drei Möglichkeiten:

1. Beide Buchstaben liegen in derselben Zeile:
Die beiden Buchstaben werden durch den jeweils nach rechts folgenden Buchstaben der Zeile ersetzt. Am Zeilenende wird am Zeilenanfang wieder begonnen.

2. Beide Buchstaben liegen in derselben Spalte:
Die beiden Buchstaben werden durch den jeweils darunter liegenden Buchstaben der Spalte ersetzt. Am Spaltenende wird am Spaltenanfang wieder begonnen.

3. Die Buchstaben liegen weder in derselben Zeile noch in derselben Spalte:
Von dem ersten Buchstaben des Buchstabenpaares wird die Zeile beibehalten und zu der Spalte des zweiten Buchstabens gewandert. Der dort befindliche Buchstabe ersetzt den ersten Buchstaben des Buchstabenpaares. Auf die gleiche Art wird auch der zweite Buchstabe chiffriert.

Daraus ergibt sich für obiges Beispiel folgende Verschlüsselung:

Klartext:	GE	HE	IM	TE	XT
Geheimtext:	MC	EI	EB	ZM	ZS

3.3.1 Entwickeln Sie zum Schlüssel "EDWARDSNOWDENNSA" die Schlüsseltabelle und verschlüsseln Sie damit den Klartext "INFORMATIK".

3.3.2 Entschlüsseln Sie den Geheimtext "PGRSICXSPQGLPOQY" mit Hilfe des, im einführenden Beispiel angegebenen, Schlüssels "PLAYFAIRCHIFFRE".

3.3.3 Klassifizieren Sie das Verschlüsselungsverfahren und nennen Sie eine effiziente Möglichkeit, einen Geheimtext (größerer Länge) ohne Kenntnis des Schlüssels zu entschlüsseln**.**

Anhang 1: Klassendokumentationen (zu Aufgabe 1)

Klasse `TAutomat`

Methoden:

`fuehreSchritteAus`

Parameter: `anzahl` (int)
Beschreibung: Ausgehend von einem vollständig weißen Spielfeld führt die Methode so viele Ausführungsschritte durch, wie im Parameter angegeben sind. Zurückgegeben wird die von der Ameise schließlich erreichte Position.
(Details über die Ausgangsposition und Blickrichtung der Ameise sollen hier vernachlässigt werden.)

Klasse `TSpielfeld`

Methoden:

`istSchwarz`
Parameter: `pos` (TPosition)
Beschreibung: Die Methode liefert den Wert `true` zurück, falls die Spielfeldzelle an der Position `pos` schwarz ist, ansonsten liefert sie den Wert `false` zurück.

`faerbeUm`
Parameter: `pos` (TPosition)
Beschreibung: Die Methode wechselt die Farbe der Zelle an der Position `pos`.

Klasse `TAmeise`

Methoden:

`gibAktuellePosition`

Parameter: -
Beschreibung: Die Methode liefert die aktuelle Position der Ameise zurück.

`bewege`
Parameter: `schwarz` (boolean)
Beschreibung: Die Methode dreht die Ameise um 90° nach links, falls `schwarz` `true` ist, ansonsten um 90° nach rechts. Anschließend wird die Ameise um ein Feld nach vorne verrückt.

Klasse `TPosition`

Methoden:

`gibX, gibY`
Parameter: -
Beschreibung: Die Methoden liefern die x- bzw. y-Koordinate der repräsentierten Spielfeldposition zurück.

Anhang 2 (zu Aufgabe 2.2): Befehlssatz des *DC*

a) Grundbefehle

Mnemo	Bedeutung
LDA	LOAD INTO ACCUMULATOR - Lade den Wert der angegebenen Speicherstelle in den Akkumulator.
STA	STORE ACCUMULATOR TO MEMORY - Speichere den Inhalt des Akkumulators an der angegebenen Speicherstelle ab.
ADD	ADD TO ACCUMULATOR - Addiere den Wert der angegebenen Speicherstelle zum Inhalt des Akkumulators.
SUB	SUBTRACT FROM ACCUMULATOR - Subtrahiere den Wert der angegebenen Speicherstelle vom Inhalt des Akkumulators.
NEG	NEGATE ACCUMULATOR - Negiere den Inhalt des Akkumulators.
INC	INCREMENT ACCUMULATOR - Erhöhe den Inhalt des Akkumulators um 1.
DEC	DECREMENT ACCUMULATOR - Erniedrige den Inhalt des Akkumulators um 1.
OUT	OUTPUT MEMORY - Gib den Wert der angegebenen Speicherstelle an die Output-Einheit. Die auszugebende Zahl erscheint in einer Zeile oberhalb des Eingabe-Fensters.
INM	INPUT TO MEMORY - Speichere die von der Input-Einheit gelesene Zahl an der angegebenen Adresse ab. Das Programm hält bei diesem Befehl an und wartet auf die Eingabe einer Zahl.
END	ENDE - Programm beenden.
DEF	DEFINE word - Beispiel: Mit `34 DEF 3` erhält die Speicherstelle mit der Adresse 34 den Wert 3 zugewiesen. Dies ist keine vom Mikroprozessor ausführbare Anweisung, sondern dient nur der Wertbelegung von Speicherstellen beim Einlesen eines DC-Programmes oder im Direktmodus.

b) Sprungbefehle

Mnemo	Bedeutung
JMP	JUMP - Unbedingter Sprung. Springe zur angegebenen Speicherstelle und fahre mit dem dort stehenden Befehl fort.
JMS	JUMP IF MINUS - Springe zur angegebenen Speicherstelle und fahre mit dem dort stehenden Befehl fort, wenn der Inhalt der Inhalt des Akkumulators negativ ist. Wenn nicht, dann fahre mit dem nächsten Befehl fort. Sprung, falls Akkumulatorinhalt < 0
JPL	JUMP IF PLUS - Sprung, falls Akkumulatorinhalt > 0
JZE	JUMP IF ZERO - Sprung, falls Akkumulatorinhalt = 0
JNM	JUMP IF NOT MINUS - Sprung, falls Akkumulatorinhalt ≥ 0
JNP	JUMP IF NOT PLUS - Sprung, falls Akkumulatorinhalt ≤ 0
JNZ	JUMP IF NOT ZERO - Sprung, falls Akkumulatorinhalt ≠ 0