Fach : Informatik
Prüfungsart: G-Niveau
Dauer : 3 Stunden
Hilfsmittel : Taschenrechner

Die Aufgaben umfassen 6 Seiten!

Aufgabe 1

1.1 Rekursion

Gegeben ist eine Funktion $f : IN \rightarrow IN$, die wie folgt rekursiv definiert ist:

$$f(n) = \begin{cases} 1, & \text{für } n = 0 \\ 2 \cdot f(n-1) + 1, & \text{für } n > 0 \end{cases}$$

1.1.1 Berechnen Sie den Wert f(3).

1.1.2 Schreiben Sie jeweils eine Delphi/Pascal-Funktion oder eine Java-Methode, die f(n)

a) gemäß der oben aufgeführten Definition

b) iterativ

berechnet.

1.2 Objektorientiertes Modellieren und Programmieren

Mengen sind Ansammlungen von Daten des gleichen Typs, in denen jeder Wert nur einmal vorkommen darf. Die Reihenfolge spielt dabei keine Rolle.

Der Abstrakte Datentyp (ADT) "Set" soll Mengen beschreiben. Die folgenden Operationen sollen ausführbar sein:

- *insert:* Fügt das übergebene Element zur Menge hinzu.
- *delete:* Entfernt das übergebene Element aus der Menge.
- *has:* Überprüft, ob das übergebene Element in der Menge enthalten ist.
- *clear:* Entfernt alle Elemente aus der Menge.
- *isEmpty:* Überprüft, ob die Menge leer ist.
- *isSubsetOf:* Überprüft, ob die Menge Teilmenge der übergebenen Menge ist.

1.2.1 Geben Sie eine Klassendefinition für eine Klasse Set an, die den ADT "Set" für integer-Zahlen realisiert. Beim Beschreiben der Methoden genügt es, den Kopf der Methoden aufzuführen.

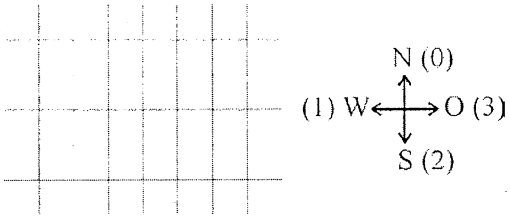
Hinweis: Sie dürfen davon ausgehen, dass die zu verwaltenden Mengen maximal 100 integer-Werte enthalten.

1.2.2 Implementieren Sie die Methoden *insert* und *isSubsetOf*.

Hinweis: Beim Schreiben dieser beiden Methoden dürfen Sie die anderen Methoden der Klasse benutzen.

Aufgabe 2:

2 Funktionen, endliche Automaten

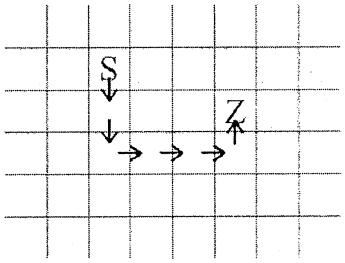


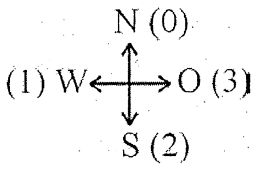
Ein Roboter bewegt sich in einer schachbrettartig angeordneten Umgebung. Er kann sich an jeder Stelle um ein Feld nach Norden, Süden, Osten oder Westen bewegen.

Gesteuert wird er, indem ihm der Pfad, entlang dessen er sich bewegen soll, als Folge von Richtungsangaben übermittelt wird. Diese Richtungsangaben werden gemäß nebenstehender Tabelle codiert.

Bewegungsrichtung	Codierung
Norden	0
Westen	1
Süden	2
Osten	3

Ein Pfad wird also als endliche Folge bestehend aus den Ziffern 0, 1, 2 und 3 angegeben. So steht z.B. die Folge 223330 für den rechts skizzierten Pfad, wobei S die Position markiert, an der der Roboter startet, und Z die Position, an der er stehen bleibt.



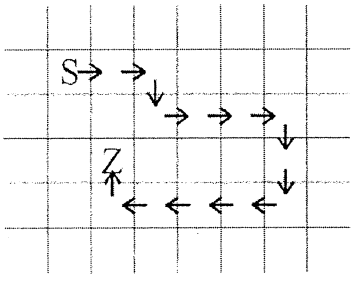


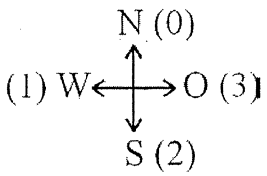
Um Übertragungsfehler möglichst zu vermeiden, wird der codierten Form des Pfades eine Prüfziffer angefügt. Diese ist eine der Ziffern 0, 1, 2 und 3 und ist so gewählt, dass die Quersumme des Pfades einschließlich der Prüfziffer durch vier teilbar ist.

Die Prüfziffer des obigen Pfades ist 3, denn 2 + 2 + 3 + 3 + 3 + 0 + 3 = 16.
Statt des eigentlichen Pfades wird also die Folge 2233303 übertragen.

Empfängt der Roboter eine solche Ziffernfolge, so überprüft er zunächst, ob die Quersumme der gesamten Folge durch vier teilbar ist. Ist dies nicht der Fall, liegt ein Übertragungsfehler vor und der Roboter sendet eine Fehlermeldung zurück. Ist sie aber durch vier teilbar, so streicht er die letzte Ziffer weg und bewegt sich entlang des übermittelten Pfads.

2.1 Geben Sie die Codierung des skizzierten Pfades einschließlich Prüfziffer an:





2.2 Der Roboter empfängt folgende Zeichenketten:

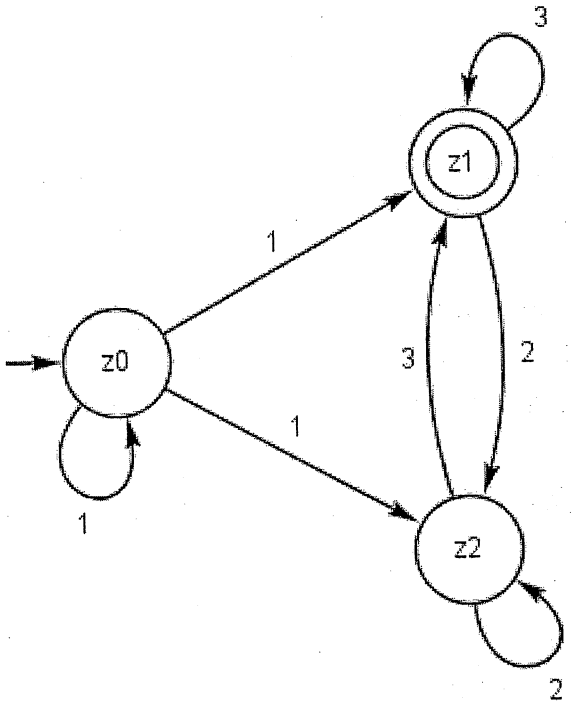
w_1 = 001122332 w_2 = 1112232333300

Überprüfen Sie jeweils, ob die Zeichenkette vom Roboter akzeptiert wird oder ein Übertragungsfehler erkannt wird.

2.3 Schreiben Sie eine Delphi/Pascal – Funktion oder eine Java – Methode, die überprüft, ob eine empfangene Zeichenkette vom Roboter akzeptiert wird.

Hinweis: Sie dürfen davon ausgehen, dass keine anderen Zeichen als 0, 1, 2 und 3 in der Zeichenkette vorkommen.

2.4 Gegeben ist der folgende Übergangsgraph eines endlichen Automaten (Akzeptors) M über dem Alphabet {1, 2, 3}:



2.4.1 Prüfen Sie, ob die Zeichenkette 1112233 vom Automat M akzeptiert wird.

2.4.2 Begründen Sie, dass der Automat M nichtdeterministisch ist.

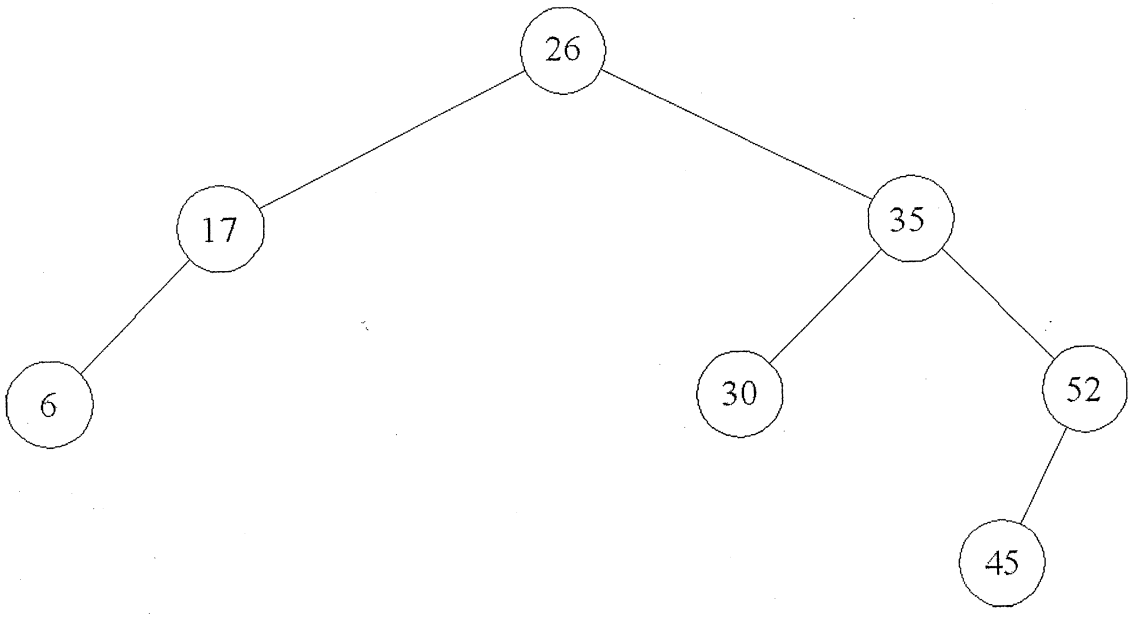
2.4.3 Erzeugen Sie mit der Teilmengenkonstruktion einen deterministischen Automaten, der die gleiche Sprache wie der Automat M akzeptiert.
Zeichnen Sie den zugehörigen Übergangsgraph.

2.5 Geben Sie den Übergangsgraph eines endlichen Automaten (Akzeptors) an, der genau die Zeichenketten akzeptiert, die der Roboter gemäß des Prüfziffer-Tests in Aufgabenteil 2.3 als korrekt bewertet.

Aufgabe 3

3.1 Geordnete binäre Bäume

Gegeben ist der folgende geordnete binäre Baum:



3.1.1 Schreiben Sie die Elemente des oben aufgeführten Baumes in Postorder-Reihenfolge auf.

Hinweis: Vorgehensweise bei der Postorder-Reihenfolge:

- Besuche den linken Teilbaum
- Besuche den rechten Teilbaum
- Besuche die Wurzel

3.1.2 Fügen Sie die folgenden Zahlen in der angegebenen Reihenfolge in den Baum ein und zeichnen Sie den am Ende vorliegenden geordneten binären Baum.

23 18 21 28 25

3.1.3 Ein binärer Baum wird als ausgeglichen bezeichnet, wenn sich alle jeweils entsprechenden linken und rechten Teilbäume in der Höhe um höchstens 1 unterscheiden.

Ein ausgeglichener geordneter Binärbaum enthalte 200 Knoten.
Geben Sie begründet an, über wie viele Knoten eine Suche nach einem Element in diesem Baum höchstens geht.

3.2 Funktionsweise von Computersystemen und Simulationsprogramm DC

Das Simulationsprogramm DC simuliert eine einfache Registermaschine (den „Didaktischen Computer"). Betrachten Sie das folgende Programm für diesen Modellrechner:

```
00 LDA 11
01 ADD 09
02 STA 11
03 LDA 09
04 SUB 10
05 STA 09
06 JNZ 00
07 OUT 11
08 END
09 DEF 5
10 DEF 1
11 DEF 0
```

Hinweis: In der Anlage finden Sie den Befehlssatz des Modellrechners.

3.2.1 Geben Sie begründet an, welche Zahl von diesem DC-Programm am Ende ausgegeben wird.

3.2.2 Schreiben Sie ein DC-Programm das zwei natürliche Zahlen einliest und deren Produkt ausgibt.

Hinweis: Das Produkt zweier Zahlen lässt sich als Summe schreiben, z.B.

$3 \cdot 5 = ((0 + 5) + 5) + 5$

3.3 RSA-Algorithmus

Für das RSA-Verfahren wurden die Primzahlen p = 17 und q = 41 gewählt.
Für das öffentliche Schlüsselpaar (e,n) gilt: e = 21.

Berechnen Sie das zu (e,n) zugehörige geheime Schlüsselpaar (d,n).

Anlage: Befehlssatz des Modellrechners

Grundbefehle

Mnemo	Intern	Bedeutung
LDA	000000	LOAD INTO ACCUMULATOR – Lade den Wert der angegebenen Speicherstelle in den Akkumulator.
STA	000001	STORE ACCUMULATOR TO MEMORY – Speichere den Akkumulatorinhalt an der angegebenen Speicherstelle.
ADD	000010	ADD TO ACCUMULATOR – Addiere den Wert der angegebenen Speicherstelle zum Akkumulatorinhalt.
SUB	000011	SUBTRACT FROM ACCUMULATOR – Subtrahiere den Wert der angegebenen Speicherstelle vom Akkumulatorinhalt.
NEG	010001	NEGATE ACCUMULATOR – Negiere den Akkumulatorinhalt.
INC	010010	INCREMENT ACCUMULATOR - Erhöhe Akkumulatorinhalt um 1.
DEC	010011	DECREMENT ACCUMULATOR - Vermindere Akkumulatorinhalt um 1.
OUT	001010	OUTPUT MEMORY – Gib den Wert der angegebenen Speicherstelle an die Output-Einheit.
INM	011100	INPUT TO MEMORY – Speichere die von der INPUT-Einheit gelesene Zahl an der angegebenen Adresse ab.
END	001011	ENDE – Programm beenden.
DEF	---	DEFINE WORD Beispiel: Mit 34 DEF 3 erhält die Speicherstelle mit der Adresse 34 den Wert 3 zugewiesen.

Sprungbefehle

Mnemo	Intern	Bedeutung
JMP	000100	JUMP – Unbedingter Sprung: Springe zur angegebenen Speicherstelle und fahre mit dem dort stehenden Befehl fort.
JNZ	000101	JUMP IF NOT ZERO Wenn der Akkumulatorinhalt ungleich Null ist: Springe zur angegebenen Speicherstelle und fahre mit dem dort stehenden Befehl fort. Sonst: Führe den nachfolgenden Befehl aus.
JZE	001001	JUMP IF ZERO
JPL	001000	JUMP IF PLUS
JNP	011011	JUMP IF NOT PLUS
JMS	010100	JUMP IF MINUS
JNM	011010	JUMP IF NOT MINUS