|  |  |  |
| --- | --- | --- |
| **Zentralabitur 2024** | **Informatik** | **Material für Prüflinge** |
| **Hinweise zur Auswahl der Aufgaben** | **gA** | **Prüfungszeit\*: 250 min** |

Die Prüfungszeit setzt sich zusammen aus 220 min Bearbeitungszeit und 30 min Auswahlzeit.

**Name:** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Klasse:** **\_\_\_\_\_\_\_**

Inhaltsverzeichnis

[Hinweise zur Auswahl der Aufgaben 3](#_Toc162335876)

[1 Operationen für Zeichenketten 5](#_Toc162335877)

[2 Operationen für dynamische Reihungen, Stapel, Schlangen und Binärbäume 6](#_Toc162335878)

[3 Datenbankabfragen 11](#_Toc162335879)

[4 Notation von Bitfolgen nach dem (7,4)-Hamming-Code 13](#_Toc162335880)

# Hinweise zur Auswahl der Aufgaben

Sie erhalten Aufgabenvorschläge aus zwei Blöcken zur Auswahl.

**Wählen Sie aus Block 1 genau eine der zwei Aufgaben zur Bearbeitung aus. Wählen Sie aus Block 2 genau zwei der vier Aufgaben zur Bearbeitung aus. Andere Kombinationen sind nicht zulässig.**

Insgesamt sind 100 Bewertungseinheiten (BE) erreichbar.  
Am Ende jeder Teilaufgabe sind die erreichbaren Bewertungseinheiten angegeben.

|  |
| --- |
| **Block 1 (50% der BE)** |
| Sie erhalten in diesem Block zwei Aufgaben zur Auswahl. Wählen Sie genau eine Aufgabe zur Bearbeitung aus. |
| Aufgabe A |
| Aufgabe B |

|  |
| --- |
| **Block 2 (50% der BE)** |
| Sie erhalten in diesem Block drei Aufgaben zur Auswahl. Wählen Sie genau zwei Aufgaben zur Bearbeitung aus. |
| Aufgabe A |
| Aufgabe B |
| Aufgabe C |

#### Hilfsmittel

Die Erläuterungen zu Operationen für Zeichenketten, Operationen für dynamische Reihungen, Stapel, Schlangen und Binärbäume, zum Sprachumfang von Datenbankabfragen, sowie zur Notation des (7,4)-Hamming-Codes sind als zugelassenes Hilfsmittel auf den Folgeseiten dokumentiert.

Die Verwendung eines Taschenrechners oder einer Formelsammlung ist nicht zulässig.

## 1 Operationen für Zeichenketten

• Bestimmen der Länge einer Zeichenkette

• Auslesen eines Zeichens an einer bestimmten Position

• Ersetzen eines Zeichens an einer bestimmten Position

• Verbinden von zwei Zeichenketten zu einer

• Prüfen des Inhalts von zwei Zeichenketten auf Gleichheit

• Lexikographisches Vergleichen von zwei Zeichenketten

Die Verwendung von darüber hinausgehenden Zeichenkettenoperationen ist nicht zulässig.

## 2 Operationen für dynamische Reihungen, Stapel, Schlangen und Binärbäume

#### Dynamische Reihung

Die Nummerierung der Elemente der dynamischen Reihung beginnt mit dem Index 0.

**DynArray()**Eine leere dynamische Reihung wird angelegt.

**isEmpty(): Wahrheitswert**Wenn die dynamische Reihung kein Element enthält, wird  
der Wert wahr zurückgegeben, sonst der Wert **falsch**.

**getItem(index: Ganzzahl): Inhaltstyp**Der Inhalt des Elements an der Position **index** wird zurückgegeben.

**append(inhalt: Inhaltstyp)**Ein neues Element mit dem übergebenen Inhalt wird am Ende der dynamischen Reihung angefügt.

**insertAt(index: Ganzzahl, inhalt: Inhaltstyp)**Ein neues Element mit dem übergebenen Inhalt wird an der Position **index** in die dynamische Reihung eingefügt. Das Element, das sich vorher an dieser Position befunden hat, und alle nachfolgenden werden nach hinten verschoben. Entspricht der Wert von **index** der Länge der dynamischen Reihung, so wird ein neues Element am Ende der dynamischen Reihung angefügt.

**setItem(index: Ganzzahl, inhalt: Inhaltstyp)**Der Inhalt des Elementes an der Position **index** wird durch den übergebenen Inhalt ersetzt.

**delete(index: Ganzzahl)**Das Element an der Position **index** wird entfernt. Alle folgenden Elemente werden um eine Position nach vorne geschoben.

**getLength(): Ganzzahl**Die Anzahl der Elemente der dynamischen Reihung wird zurückgegeben.

#### Stapel

**Stack()**Ein leerer Stapel wird angelegt.

**isEmpty(): Wahrheitswert**Wenn der Stapel kein Element enthält, wird der Wert **wahr** zurückgegeben, sonst der Wert **falsch**.

**top(): Inhaltstyp**Der Inhalt des obersten Elements des Stapels wird zurückgegeben, das Element aber nicht entnommen.

**push(inhalt: Inhaltstyp)**Ein neues Element mit dem übergebenen Inhalt wird oben  
auf den Stapel gelegt.

**pop(): Inhaltstyp**Das oberste Element des Stapels wird entnommen. Der Inhalt dieses Elements wird zurückgegeben.

#### Schlange

**Queue()**Eine leere Schlange wird angelegt.

**isEmpty(): Wahrheitswert**Wenn die Schlange kein Element enthält, wird der Wert **wahr** zurückgegeben, sonst der Wert **falsch**.

**head(): Inhaltstyp**Der Inhalt des vordersten Elements der Schlange wird zurückgegeben, das Element aber nicht entnommen.

**enqueue(inhalt: Inhaltstyp)**Ein neues Element mit dem angegebenen Inhalt wird am Ende an die Schlange angehängt.

**dequeue(): Inhaltstyp**Das vorderste Element wird entnommen. Der Inhalt dieses Elements wird zurückgegeben.

#### Binärbaum

**BinTree()**Ein Baum wird erzeugt. Der Baum besitzt keine Teilbäume.  
Die Wurzel besitzt keinen Inhaltswert.

**BinTree(inhalt: Inhaltstyp)**Ein Baum wird erzeugt. Der Baum besitzt keine Teilbäume.  
Die Wurzel erhält den übergebenen Inhalt als Wert.

**hasItem(): Wahrheitswert**Wenn die Wurzel des Baums einen Inhaltswert besitzt, wird  
der Wert **wahr** zurückgegeben, sonst der Wert **falsch**.

**getItem(): Inhaltstyp**Die Operation gibt den Inhaltswert der Wurzel des Baumes zurück.

**setItem(inhalt: Inhaltstyp)**Die Wurzel des Baums erhält den übergebenen Inhalt als Wert.

**deleteItem()**Die Operation löscht den Inhaltswert der Wurzel des Baums.

**isLeaf(): Wahrheitswert**Wenn der Baum keine Teilbäume besitzt, die Wurzel des Baums also ein Blatt ist, wird der Wert **wahr** zurückgegeben, sonst der Wert **falsch**.

**hasLeft(): Wahrheitswert**Wenn der Baum einen linken Teilbaum besitzt, wird der Wert **wahr** zurückgegeben, sonst der Wert **falsch**.

**getLeft(): Binärbaum**Die Operation gibt den linken Teilbaum zurück.

**setLeft(b: Binärbaum)**Der übergebene Baum wird als linker Teilbaum gesetzt.

**deleteLeft()**Die Operation löscht den linken Teilbaum.

**hasRight(): Wahrheitswert**Wenn der Baum einen rechten Teilbaum besitzt, wird der Wert **wahr** zurückgegeben, sonst der Wert **falsch**.

**getRight(): Binärbaum**Die Operation gibt den rechten Teilbaum zurück.

**setRight(b: Binärbaum)**Der übergebene Baum wird als rechter Teilbaum gesetzt.

**deleteRight()**Die Operation löscht den rechten Teilbaum.

## 3 Datenbankabfragen

#### SELECT-Anweisung:

**SELECT** [DISTINCT | ALL] \* | spalte1[AS alias1], spalte2 [AS alias2],  
…, spalten [AS aliasn]

**FROM** tabelle1, tabelle2, …, tabellem

[**WHERE** bedingung1 (AND | OR)  
bedingung2 … (AND | OR) bedingungk]

[**GROUP BY** spalte1, spalte2, … , spaltep

[**HAVING** gruppenBedingung1 (AND | OR)  
 gruppenBedingung2 … (AND | OR)  
 gruppenBedingungs] ]

[**ORDER BY** spalte1 [ASC | DESC],  
 spalte2 [ASC | DESC],…,  
 spaltet [ASC | DESC] ]

[**LIMIT** anzahl]

Angaben in eckigen Klammern sind optional. Spalten können Attribute, Aggregatfunktionen oder Berechnungen sein.  
Bei **GROUP BY** und **ORDER BY** ist auch die Angabe eines Alias möglich.

**Operatoren für Berechnungen:** +, -, \*, /

**Operatoren für Vergleiche in Bedingungen:** =, != (ungleich), >, <, >=, <=, **NOT**, **LIKE** (mit den Platzhaltern \_ und %), **BETWEEN**, **IN**, **IS NULL**

**Aggregatfunktionen:** **AVG**, **COUNT**, **MAX**, **MIN**, **SUM**

## 4 Notation von Bitfolgen nach dem (7,4)-Hamming-Code

Die Daten- und Prüfbits im (7,4)-Hamming-Code werden  
in der Reihenfolge **p0 p1 d0 p2 d1 d2 d3** notiert.

Die Kontrollgruppen sind entsprechend der folgenden Abbildung zusammengesetzt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Datenbits | | | |
| Prüfbit | **d0** | **d1** | **d2** | **d3** |
| **p0** | x | x | - | x |
| **p1** | x | - | x | x |
| **p2** | - | x | x | x |