

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EKSAMEN/TOETS**  **EXAMINATION/TEST:** | **Semester test 2017 (2)** | **KWALIFIKASIE/**  **QUALIFICATION:** | **BSc** | |
| **MODULEKODE/**  **MODULE CODE:** | **ITRW222** | | **TYDSDUUR/**  **DURATION:** | **2 hours** |
| **MODULEBESKRYWING/**  **MODULE DESCRIPTION:** | **Datastrukture/**  **Data Structures** | | **MAKS/**  **MAX:** | **60** |
| **EKSAMINATOR(E)/**  **EXAMINER(S):** | **Prof. R Goede** | | **DATUM/**  **DATE:** | **09/10/2017** |
| **MODERATOR/**  **MODERATOR:** |  | | **TYD/TIME:** | **10:00** |

**Vraag 1/ *Question 1* (25)**

|  |  |
| --- | --- |
| Gebruik die *τ -* notasie om die looptyd van die volgende programlyne te bepaal. (15) | *1.1 Use* ***τ******-*** *notation to determine* ***the running time*** *of the following program lines (15)* |
| 1. for ( int i=1; i<=n+1; i++) {  2. b= 1 + Fibo(arr[i]); } // Fibo is a method | |

|  |  |  |
| --- | --- | --- |
| 1. 1.2 Bepaal die looptyd van al drie dele van lyn 8 in konteks van hierdie programdeel. Jy hoef nie die uitdrukkings te vereenvoudig nie. Maak gebruik van die vereenvoudigde model. (8) | | *1.2 . Determine the running time of all three parts of line 8 in context of this program segment. You need not simplify the expressions. Use the simplified model. (8)* |
| 1 public class Question1.2  2 {  3 public static int numbers (int n)   1. { 2. int prod = 1; 3. for (int i=1; i<n; i++ ) 4. { 5. for ( int j=1; j<=i+1; ++j) 6. prod \*=j; 7. } 8. return prod; 9. } 10. } | | |
| 1.3 Gee die definisie van O(n) (2) | *1.3.* Give the definition for: O(n) (2) | |

**Vraag 2 / *Question 2* (5)**

|  |  |
| --- | --- |
| 2. Aanvaar die volgende kode bestaan: Ontwerp ’n metode vir die klas Stack genaamd: *pop()* in Java (5) | 2. Assume the following code exists.  Design a method for the class Stack called: *pop()* in Java. (5) |
| public class Stack  { private Listing[] data;  private int top;  private int size;  public Stack( )  { top = -1;  size = 100;  data = new Listing[100];  } | |

**Vraag 3/ *Question 3* (25)**

|  |  |
| --- | --- |
| 3 Bestudeer die volgende klas: | *3 Study the following class:* |
| public class SLL<T extends Comparable<? super T>>  {  private Element<T> head; // list header  private Element<T> tail;    public SLL()  { head = null;  tail = null;}  …  public class Element<T1 extends Comparable<? super T>>  {  private T1 data;  private Element<T1> next;  public Element(T1 param)  {  data = param;  }  }// end of inner class Node  } | |
| 3.1 Skryf ‘n metode in Java vir die klas SLL genaamd deleteElement wat ‘n spesifieke element uit die lys verwyder – VOLGENS ADRES VAN DIE ELEMENT EN NIE DIE WAARDE NIE. Wenke: die parameter moet ‘n adres wees: boolean deleteElement (Element<T> item)1. Die proses is bykans presies dieselfde as om volgens waarde uit te vee.2. Adresse kan direk sonder compareTo() met mekaar vergelyk word.(8) | *3.1 Write a method in Java for the class SLL called DeleteElement that removes a specific element ACCORDING TO THE ADDRESS OF THE ELEMENT AND NOT THE VALUE. The parameter should be an address: boolean deleteElement(Element<T> item).*  *1. The process is almost exactly the same as to delete by value.*  *2. Addresses may be compared directly without the use of compareTo().*  *(8)* |
| 3.2 Ontwerp ‘n metode vir die klas SLL genaamd:*deDup()* om duplikate uit die geskakelde lys te verwyder.*Roepende lys = {3,1,5,8,6,5,4,3,7,1,1,8}* *NA deDup():*  *Roepende lys = {3,1,5,8,6,4,7}*  Jy moet die kode gee vir al die bestaande SLL metodes wat jy wil gebruik, behalwe toString() en DeleteElement.  Jy mag nie ‘n tydelike lys gebruik nie. | *3.2 Design a method for the class SLL called:*  *deDup() to remove duplicates from the linked list..* *Calling list = {3,1,5,8,6,5,4,3,7,1,1,8}* *AfterDeDup():*  *Calling list = {3,1,5,8,6,4,7}*  *You have to give all the code for existing SLL methods you want to use, except toString() and deleteElement().*  *You many not use a temporary list.* |
| 3.2.1 Teken ‘n geskakelde lys om jou met die algotirme-ontwerp te help.  Skryf die algemene en die spesiale gevalle vir die probleem in Afrikaans neer. Gee ‘n kort beskrywing van die nodige aksie vir elkeen van die gevalle. (5) | *3.2.1 Draw a linked list to help you to design the algorithm.*  *Write down the general and all the special cases for the problem in English. Give a short description of the required action for each of these cases. (5)* |
| 3.2.2. Skryf die metode ***deDup(…)*** in Java. (12) | *3.2.2 Write the method deDup(…) in Java. (12)* |
| 3.2.3 Skryf ‘n drywer program om die metode deeglik te toets. (5) | *3.2.3 Write a driver program to test the method thoroughly. (5)* |
| 3.2.4 BONUS vraag: skryf ‘n konstruktor vir die SLL klas wat ‘n skikking vol generiese waardes ontvang en die waardes in die geskakelde lys plaas. (5) | *3.2.4 BONUS question: Write constructor for the SLL class that receives an array of generics as parameter and loads values into the linked list. (5)* |

**Vraag 4/ *Question 4* (5)**

|  |  |
| --- | --- |
| 4.1 Verduidelik in jou eie woorde hoe ‘n “hash” table data stoor (3) | *4.1 Explain in your own words how a hash table stores data. (3)* |
| 4.2 Wat is ‘n perfekte “hashing” – funksie? (2) | *4.2 What is a perfect hashing function? (2)* |