

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **EKSAMEN/TOETS**  **EXAMINATION/TEST:** | **Semester test 2017 MEMO** | **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:** | **26/09/17** |
| **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 | |
| 1a. tfetch + tstore √  1b. (3tfetch + t<+ t+)√ (n+2) √  1c. (2tfetch √+ t+ √+ tstore √)(n+1) √  2. (4tfetch √√+ t+ √+ tstore √+ t[.]√+tstore√+TFibo√ +T call )(n+1) √ | |

|  |  |  |
| --- | --- | --- |
| 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. } | | |
| 8a 2(n-1)√ √  8b √√√  8c √√√ | | |
| 1.3 Gee die definisie van O(n) (2) | *1.3.* Give the definition for: O(n) (2) | |
| Consider a function f(n) that is non-negative for all inters n>=0. We say that “f(n) is big oh g(n)” which we write f(n) = O(g(n)), if there exists an inter n0 and a constant c> 0 so that for all integers n>=n0, f(n) <= c g(n). √√ | | |

**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];  } | |
| Ontwerp ’n metode vir die klas Stack genaamd: *push()*  public Listing pop() √  int toplocation;  { if(top == -1) √  return null; // \*\* underflow error \*\*√  else  {  toplocation = top;  top = top-1; √  return data[toplocation]; √  }  } | |

**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 append() wat ‘n nuwe element aan die einde van die lys byvoeg. (5) | *3.1 Write a method in Java for the class SLL called append that adds a new element at the back of the list (5)* |
|  | |
| 3.2 Ontwerp ‘n metode genaamd:*filter( …)* om sekere elemente uit ‘n geskakelde lys te verwyder. Jou metodoe ontvang ‘n GESKAKELDE LYS wat PRESIES uit TWEE element bestaan (jy moet dit toets) as parameter. Filter() moet al die elemente uit die roepende lys uit haal wat KLEINER as die eerste PARAMETER LYS item is en wat GROTER AS die tweede PARAMETER LYS item is: bv. *Roepende lys = {3,1,5,8,6,5,4,3,7,1,8}*  *parameter lys= {4,7}*  *NA filter:*  *Roepende lys = {5,6,5,4,7}*  Jy moet die kode gee vir al die bestaande SLL metodes wat jy wil gebruik, behalwe toString() | *3.2 Design a method for the class called:*  **filter(…)** *to remove certain elements from the list. Your method receives a LINKED LIST containing PRECISELY TWO element (you need to verify this) as parameter. Filter() should remove all the elements in the calling list which are SMALLER than the first PARAMETER LIST item and LARGER THAN the second PARAMETER LIST item: eg.*  *calling List= {3,1,5,8,6,5,4,3,7,1,8}*  *parameter list= {4,7}*  *AFTER filter:*  *Calling List = {5,6,5,4,7}*  You have to give all the code for existing SLL methods you want to use, except toString() |
| 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)* |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | *Calling list* |  | *A200* |  | *B145* |  | *C001* |  | *DDDD* | | *Head = A200* |  | *5* |  | *7* |  | *5* |  | *2* | | *Tail = DDD* |  | *B145* |  | *C001* |  | *DDDD* |  | *null* |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  | |  |  |  |  |  |  |  |   *General – traverse calling list with TWO while loops*  *The outer loop starts at head and set a tmp ptr to the first node – the second loop then runs through the rest of the SLL and compares the data of each element to the data of the first element – if they are similar – that node should be deleted by Address and not by value – by value will delete the first occurrence and you want the subsequent versions deleted.*  *Special*  *List can be empty - return*  *Tail can be a duplicate – set new tail*  *No duplicates – no special action* | |
| 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 W*rite 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) |
| When a mapping is done between the KEY FIELD VALUE and storage index TO PREVENT SEQUENCIAL search of an array = A formula is used to map the key field to the memory address | |
| 4.2 Wat is ‘n perfekte “hashing” – funksie? (2) | 4.2 What is a perfect hashing function? (2) |
| When the hashing algorithm returns a unique value for the address. | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Marking scheme Question 3** | |  |  |
| **Question 1** |  |  |  |
| Create ptr and prevPtr | 0 | 1 |  |
| Test for empty list | 0 | 1 |  |
| If ptr==head | 0 | 1 |  |
| While ptr!= null && ptr!=item | 0 | 1 |  |
| Advance ptrs | 0 | 1 |  |
| update tail if empty | 0 | 1 |  |
| Set ptrs to delete element | 0 | 1 |  |
| update tail if not empty | 0 | 1 |  |
|  |  |  |  |
|  |  | **8** |  |
| **Question 2** |  |  |  |
| Diagram – calling list | 0 | 1 | 2 |
| General case description | 0 |  | 1 |
| Special case description | 0 |  | 1 |
|  |  |  |  |
| Heading of method | 0 | 1 | 2 |
| 3 ptrs for 2 whiles | 0 | 1 | 2 |
| Outer while | 0 | 1 |  |
| Inner while start | 0 | 1 |  |
| Compare data of 2 ptrs | 0 | 1 | 2 |
| Correct compareTo | 0 | 1 |  |
| Delete by address | 0 | 1 |  |
| Advance inner loop pointer | 0 | 1 |  |
| Advance outer loop pointer | 0 | 1 |  |
|  |  | **12** |  |
| **Question 3** |  |  |  |
| New SLL list | 0 | 1 |  |
| Test method for empty list | 0 | 1 |  |
| Add elements into list | 0 | 1 |  |
| Test general case | 0 | 1 |  |
| Appropriate printing methods | 0 | 1 |  |
| **-2 if numbers are added to lists rather than objects** |  | **5** |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| **TOTAL** |  | 25 |  |