Technical Test

DEV - D

Date: 17/12/2024 City/State: Recife / PE

Course: Computer Science Educational Institution: CESAR School

Course Duration (in years): 4 Current Semester: 5th Graduation Year (expected): 2026

Availability to work: ☐ 20h [✓ 30h] ☐ 40h Estimated Start Date: January / February

Instructions:

This test consists of 8 multiple choice questions, 1 algorithm implementation and 1 non-technical question. The algorithm is worth 60% of the total score. The non-technical question must be answered in Portuguese.

You may use any blank space on this test as a draft.

Use the table below to record your answers.

Good luck!

Answer Sheet

	1	2	3	4	5	6	7	8
A								x
B				x		x	x	
C	x	x	x					
D					x			

Question 1

Given:

```
1.   public class A {
2.   public void print(){ System.out.println("x"); } 3.
     }
4.   public class B extends A {
5.   public void print(){ System.out.println("y"); } 6.
     }
7.   public class Main{
8.       public static void main(String[] args) {
9.           B test = new A();
10.          test.print();
11.      }
12.  }
```

What’s the output?

A. x

B. y

C. Runtime or Compilation error on line 9

D. Runtime or Compilation error on line 10

Alternative A is Incorrect because assigning an A object to a B reference causes a compilation error.

Question 2

Given:

```
1.   //******************************
2.   // file A.java
3.   //******************************
4.   package a;
5.   public class A {
6.       private int x;
7.       protected int y;
8.       public int m1() {return x;}
9.   }
10. //******************************
11. // file B.java
12. //******************************
13.  package b;
14.  import a.A;
15.  public class B extends A {
16.      private int z;
17.      public void m2(A a){
18.          z = y;
19.          z = a.x;
20.          z = a.m1();
21.      }
22. }
```

Consider the following statements:

I. Line 18 is not valid because y is protected on A

II. Line 19 is not valid because x is private

III. Line 20 is valid because m1() is public

A. Only I and II are correct

B. Only I and III are correct

C. Only II and III are correct

D. I, II and III are correct

Alternative a is Incorrect. y is protected, so it is accessible in the subclass.
Option b is Incorrect. Line 19 fails because x is private.
Alternative d: is also Incorrect because line 19 still fails.

Question 3

What are the major elements in an object model?

A. Abstraction, encapsulation and persistence "Persistence" is not a fundamental object-oriented concept.
B. Hierarchy, persistence and typing "Typing" and "Persistence" are supplementary concepts.
C. Abstraction, encapsulation and hierarchy
D. Hierarchy, concurrency and typing

Question 4

Suppose the keys `{1,2,3,4,6,9}` are inserted into a hash table with collisions resolved by chaining. Let the table have 3 slots, and the hash function be `h(k) = k % 3`, that is, `h(k)` returns the remainder of the Euclidean division of k by 3. How will the keys be organized into this hash table?

A. 1->4	B. 3->6->9	C. 3->4	D. 1
2->6	1->4	1->6	2->4
3->9	2	2->9	3->6->9

Alternative A: Incorrect distribution. The hash function (k % 3) is not applied correctly.
Alternative C: Incorrect order. The positions of the keys are mismatched.
Alternative D: Incorrect because the chaining order is not represented properly.

Question 5

Consider the following statements:

I. A Binary Tree is a tree data structure in which each node has at most two child nodes, usually distinguished as "left" and "right", and a tree with *n* nodes has exactly *n*−1 branches.
II. A Hash Map is a data structure in which, if there’s no collision among the keys, you can always find an element in O(1) time, even in the worst case.
III. In a doubly linked list data structure, each item is linked to both the previous and the next items in the list, allowing easy access of list items backwards as well as forwards.

A. Only II and III are correct
B. Only II is correct
C. Only III is correct
D. I, II and III are correct

Alternative A is incorrect. "Binary Tree" ensures n nodes have 1 edges; the statement is valid.
Alternative B is incorrect. hash map access time is O(1) if there are no collisions.
Alternative C is incorrect. doubly linked list navigation in both directions is correct.

Question 6

In the following code, assume that Queue is not thread-safe, there is more than one Producer thread and more than one Consumer thread running and this program is crashing on runtime. In order to fix the code below how should you fill in lines (1), (2), (3) and (4)?

Global variables

```
Queue q;
(1)
```

Producer thread	**Consumer thread**

```
runProducer(){
  while(true){
    item = new item();
    (2)
    if (q is not full){
      q.enqueue(item);
      (3)
    }
    (4)
  }
}
```
	```
	runConsumer(){
	  while(true){
	    (2)
	    if (q is not empty){
	      item = q.dequeue();
	      (3)
	    }
	    (4)
	  }
	}
	```

A. (1)
   (2)
   (3) `if(Consumer) sleep(1); else sleep(2);`
   (4)

B. (1) `mutex m;`
   (2)
   (3) `m.lock();`
   (4) `m.unlock();`

Incorrect. Using sleep does not resolve race conditions in multithreaded programs.

C. (1) `semaphore guard;`
   (2) `wait(guard);`
   (3)
   (4) `signal(guard);`

Incorrect. Semaphores are not ideal for simple mutual exclusion.

D. Alternatives A, B and C are all correct. Incorrect. Sleep alone cannot guarantee synchronization.

Question 7

Considering the following tables and data information, what would be the correct result of the SQL command below?

Salesperson			
ID	**Name**	**Age**	**Salary**
1	Abe	61	140,000
2	Bob	34	44,000
5	Chris	34	40,000
7	Dan	41	52,000
8	Ken	57	115,000
11	Joe	38	38,000

Customer			
ID	Name	City	Industry_Type
4	Samsonic	Pleasant	G
6	Panasung	Oaktown	N
7	Samony	Jackson	N
9	Ornange	Hayward	G
8	Hepoul	Cupertino	I

Orders				
Number	**Order_Date**	**cust_id**	**salesperson_id**	**Amount**
10	8/2/2010	4	2	540
20	5/6/2012	9	7	150
30	3/12/2012	8	5	1,500
40	1/30/2013	4	8	1,800
50	7/14/2009	9	1	460
60	1/29/2012	7	2	2,400
70	2/3/2012	6	7	600
80	4/1/2013	8	2	2,300
90	3/2/2012	6	7	720

```
SELECT Salesperson.Name from Salesperson
WHERE Salesperson.ID NOT IN(
    SELECT Orders.salesperson_id FROM Orders
    INNER JOIN Customer ON Orders.cust_id = Customer.ID
    WHERE Customer.Name = 'Samsonic')
AND Salesperson.ID IN
    (SELECT DISTINCT Orders.salesperson_id FROM Orders);
```

A. Abe
   Chris
   Dan

B. Abe
   Bob
   Chris
   Dan

C. Abe
   Chris
   Dan
   Joe

D. Bob
   Ken

Option A leaves out Bob, who did sell to a G-type customer (Samsonic).
Option C incorrectly adds Joe, who sold only to an N-type customer, not a G-type customer.
Option D only includes Bob and Ken. Ken made no sales to G-type customers, and it omits Abe, Chris, and Dan, who did.

Question 8

Given this output on a Linux terminal:

```
$ cat unix_like_systems.txt
MINIX operating system
UNIX operating system
Linux operating system
MINIX and UNIX operating system
MINIX and Linux operating system
UNIX and Linux operating system
```

What will be the correct result of the command below?

```
$ cat unix_like_systems.txt | grep UNIX | sort
```

A.
```
MINIX and UNIX operating system
UNIX and Linux operating system
UNIX operating system
```

B.
```
operating system UNIX
and MINIX operating system UNIX
and Linux operating system UNIX
```

C.
```
and MINIX operating system UNIX
and Linux operating system UNIX
operating system UNIX
```

D.
```
UNIX operating system
MINIX and UNIX operating system
UNIX and Linux operating system
```

Alternative B is incorrect because it does not apply the sort step, so the results remain in the original file order.
Alternative C is incorrect because the grep filter does not produce the expected sorted output.
Alternative D is incorrect because the order of the results does not match the proper sort.

Infinite Coins

Given an infinite number of quarters (25 cents), dimes (10 cents), nickels (5 cents) and pennies (1 cent), write a method **makeChange()** to calculate and return a Set containing the ways of representing n cents using those coins. Each element of the returned Set should represent the number of coins to compose the entry value, an array like this: `[quarters, dimes, nickels, pennies]`.

The method **makeChange()** can use a Set data structure to store each representation of n cents, and then, return it. A **Set** is a collection that contains no duplicate elements and the order of elements is irrelevant. Consider the following interface defined for Set:

Method signature	Method description
`boolean add(Element e)`	Adds the specified element to this set if it is not already present (optional operation).
`boolean addAll(Set s)`	Adds all elements from s that are not already present in this set.
`boolean contains(Element e)`	Returns true if this set contains the specified element.
`boolean equals(Set s)`	Compares the specified set s with this set for equality.
`Iterator<Element> iterator()`	Returns an iterator over the elements in this set.
`boolean remove(Element e)`	Removes the specified element from this set if it is present (optional operation).
`int size()`	Returns the number of elements in this set (its cardinality).
`Element[] toArray()`	Returns an array containing all of the elements in this set.

Table: Set interface

Input example:

```
n=12
```

Output for the given example:

```
[[0,0,0,12], [0,0,1,7], [0,0,2,2], [0,1,0,2]]*
```

* this is the content of the **Set** which should be returned by the function.

Your proposed solution can be written in **pseudo-code** or any well-known language (C, C++, Java, etc) and you are free to implement any auxiliary functions. Besides, write down a comment to the implemented function, explaining how your function will work like the one below.

```
/**
* The function below will ...
* - Obtain the input
* - Iterate over the elements
* …
* - Print the output and return ...
*/
```

Algorithm Solution

```
import java.util.Scanner;

public class CoinProgram {

    // A simple custom Set-like class to store unique combinations of coins.
    static class MySet {
        private int[][] elements; // Stores each combination.
        private int size;         // How many combinations are currently stored.

        public MySet() {
            // Initialize with capacity 10.
            elements = new int[10][];
            size = 0;
        }

        /**
         * Adds a new combination to the set if not already present.
         * @param combination an array [q, d, n, p] representing one combination
         * @return true if added, false if already exists
         */
        public boolean add(int[] combination) {
            if (contains(combination)) {
                return false;
            }
            ensureCapacity();
            elements[size] = combination;
            size++;
            return true;
        }

        /**
         * Checks if a given combination already exists in the set.
         * @param combination the combination to check
         * @return true if it exists, false otherwise
         */
        private boolean contains(int[] combination) {
            for (int i = 0; i < size; i++) {
                if (areEqual(elements[i], combination)) {
                    return true;
                }
            }
            return false;
        }

        /**
         * Checks if two combinations are equal by comparing each element.
         * @param a first combination
         * @param b second combination
         * @return true if identical, false otherwise
         */
        private boolean areEqual(int[] a, int[] b) {
            if (a.length != b.length) return false;
            for (int i = 0; i < a.length; i++) {
                if (a[i] != b[i]) return false;
            }
            return true;
        }

        /**
         * Ensures the internal array can store at least one more combination.
         * If not, it doubles the array size.
         */
        private void ensureCapacity() {
            if (size >= elements.length) {
                int newCapacity = elements.length * 2;
                int[][] newArr = new int[newCapacity][];
                for (int i = 0; i < size; i++) {
                    newArr[i] = elements[i];
                }
                elements = newArr;
            }
        }

        /**
         * Returns all stored combinations as a 2D
array.
         * @return a 2D int array of all combinations
         */
        public int[][] toArray() {
            int[][] result = new int[size][];
            for (int i = 0; i < size; i++) {
                result[i] = elements[i];
            }
            return result;
        }
    }

    /**
     * Computes all combinations of quarters(25c),
dimes(10c), nickels(5c), and pennies(1c)
     * that sum up to n cents. It returns these
combinations in a custom MySet to avoid
duplicates.
     * @param n the total amount in cents to make
change for
     * @return a MySet containing arrays [q, d, n, p]
     */
    public static MySet makeChange(int n) {
        MySet result = new MySet();
        for (int q = 0; q <= n / 25; q++) {
            int afterQuarters = n - q * 25;
            for (int d = 0; d <= afterQuarters / 10; d++) {
                int afterDimes = afterQuarters - d * 10;
                for (int ni = 0; ni <= afterDimes / 5; ni++) {
                    int afterNickels = afterDimes - ni * 5;
                    int p = afterNickels;
                    if (q * 25 + d * 10 + ni * 5 + p == n) {
                        int[] combo = {q, d, ni, p};
                        result.add(combo);
                    }
                }
            }
        }
        return result;
    }

    /**
     * Main method:
     * - Reads an integer (cents) from the user
     * - Calls makeChange to find all combinations
     * - Prints each combination
     */
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the amount in cents: ");
        int n = scanner.nextInt();
        scanner.close();

        MySet ways = makeChange(n);
        int[][] arr = ways.toArray();

        System.out.println("Ways to make " + n + "
cents:");
        for (int[] combination : arr) {
            System.out.println("[" + combination[0] + ","
                               + combination[1] + ","
                               + combination[2] + ","
                               + combination[3] + "]");
        }
    }
}
```

Qual a disciplina que você mais gostou de cursar na faculdade e por quê? (Responder em português)

A disciplina que mais gostei de cursar na universidade até o momento foi Infraestrutura de Comunicação, também conhecida como Redes ou Infraestrutura de Redes. A razão de essa ser minha matéria favorita é que, desde pequeno, sempre me identifiquei como uma pessoa extremamente curiosa, alguém com fome de aprender tópicos novos e entender como as coisas funcionam. Foi exatamente isso que essa disciplina me proporcionou.

Compreender o funcionamento dos protocolos de rede, as camadas dos modelos OSI e híbrido, e como ocorrem as comunicações via internet, com todas as etapas envolvidas, tem sido uma experiência incrível. Ter a oportunidade de estudar e entender a tecnologia com a qual temos tanto contato no dia a dia foi transformador. Além disso, a realização de um projeto que utilizava sockets em Python para implementar a comunicação entre cliente e servidor consolidou ainda mais meu interesse pela área.

Por todos esses motivos, Infraestrutura de Comunicação se tornou minha disciplina favorita, pois despertou minha curiosidade e me proporcionou aprendizados práticos e teóricos fundamentais.