Technical Test

DEV - D

Date: 24/01/2025 City/State: Contagem/MG

Course: Sistemas de Informação Educational Institution: Puc Minas - Contagem

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

Availability to work: ☐ 20h ☒ 30h ☒ 40h Estimated Start Date: 17:43

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	X
B				X		X		
C	X		X					
D		X			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

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

Question 3

What are the major elements in an object model?

A. Abstraction, encapsulation and persistence
B. Hierarchy, persistence and typing
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
2->6
3->9

B. 3->6->9
1->4
2

C. 3->4
1->6
2->9

D. 1
2->4
3->6->9

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

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();`

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

D. Alternatives A, B and C are all correct.

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

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
```

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

```
using System;
using System.Collections.Generic;

public class InfiniteCoins
{
    /**
     * A função abaixo calcula todas as combinações possíveis de moedas (quarters,
dimes, nickels e pennies)
     * que somam o valor n em centavos.
     * - A função começa com a chamada para 'MakeChangeRecursive', que é recursiva e
tenta todas as combinações
     *   de moedas possíveis.
     * - Quando uma combinação válida é encontrada (que soma o valor n), ela é
adicionada ao conjunto.
     * - O conjunto é retornado no final, contendo todas as combinações únicas.
     */
    public static HashSet<int[]> MakeChange(int n)
    {
        // Cria um HashSet para armazenar as combinações únicas
        HashSet<int[]> result = new HashSet<int[]>(new IntArrayComparer());

        // Chama a função recursiva para explorar todas as combinações possíveis
        MakeChangeRecursive(n, 0, new int[4], result);

        return result;
    }

    /**
     * Função recursiva para gerar todas as combinações possíveis de moedas
     */
    private static void MakeChangeRecursive(int amountLeft, int coinIndex, int[]
currentCombination, HashSet<int[]> result)
    {
        // Caso base: se o valor restante for 0, adicionamos a combinação ao resultado
        if (amountLeft == 0)
        {
            result.Add((int[])currentCombination.Clone());
            return;
        }

        // Caso recursivo: tentamos usar diferentes quantidades de moedas a partir do
índice coinIndex
        if (coinIndex < 4)
        {
            // Calcula o número máximo de moedas do tipo 'coinIndex' que podem ser
usadas
            int maxCoins = amountLeft / GetCoinValue(coinIndex);

            // Tentamos de 0 até maxCoins moedas do tipo coinIndex
            for (int count = 0; count <= maxCoins; count++)
```

```
            {
                currentCombination[coinIndex] = count;
                MakeChangeRecursive(amountLeft - count * GetCoinValue(coinIndex),
coinIndex + 1, currentCombination, result);
            }

            // Limpa o valor atual para explorar outras opções
            currentCombination[coinIndex] = 0;
        }
    }

    // Função para obter o valor de uma moeda baseado no índice
    private static int GetCoinValue(int index)
    {
        switch (index)
        {
            case 0: return 25; // quarter
            case 1: return 10; // dime
            case 2: return 5;  // nickel
            case 3: return 1;  // penny
            default: throw new ArgumentException("Índice de moeda inválido");
        }
    }

    /**
     * Comparador para garantir que o HashSet armazene combinações únicas de int[].
     * O método Equals e GetHashCode são usados para garantir que as combinações sejam
tratadas como únicas
     * baseado no conteúdo dos arrays (e não no endereço de memória).
     */
    public class IntArrayComparer : IEqualityComparer<int[]>
    {
        public bool Equals(int[] x, int[] y)
        {
            if (x == null || y == null)
                return false;

            // Verifica se ambos os arrays têm os mesmos valores em cada posição
            for (int i = 0; i < x.Length; i++)
            {
                if (x[i] != y[i])
                    return false;
            }
            return true;
        }

        public int GetHashCode(int[] obj)
        {
            // Gera um código hash com base nos valores do array
            int hash = 17;
            foreach (int value in obj)
            {
                hash = hash * 23 + value;
```

```
            }
            return hash;
        }
    }

    // Exemplo de uso
    public static void Main()
    {
        int n = 12;
        HashSet<int[]> combinations = MakeChange(n);

        // Imprime o número total de maneiras de representar n centavos
        Console.WriteLine($"Número total de maneiras de representar {n} centavos:
{combinations.Count}");

        // Imprime cada combinação no formato desejado
        foreach (var combination in combinations)
        {
            Console.WriteLine($"[{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 até agora foi ATP no primeiro período, foi onde aprendi a lógica de programação na linguagem C#, e na minha opinião foi difícil entender o “básico” por não ter muita noção, mas fui me acostumando e entendendo como tudo funcionava.