

[Área personal](#) / [Mis cursos](#) / [\[1-2022\]_INF220-SD](#) / [Examen Parcial 2](#) / [Examen Parcial 2](#)

Comenzado el Thursday, 28 de July de 2022, 12:21
Estado Finalizado
Finalizado en Thursday, 28 de July de 2022, 12:29
Tiempo empleado 8 minutos 11 segundos
Calificación 100,00 de 100,00
Comentario - Respuesta correcta del examen

```
public clsLista AddS(int i, int x)
{
    if (!Llena(i))                // Se adicionan elementos si no esta llena
        if (Vacía(i))             // Si esta vacía la lista i se adiciona el primer elemento
        {
            listaS[i].AddPrimero(x);
            C[i] = C[i] + 1;        // incrementa en uno el contador de elemento de la lista i
        }
        else                      // Sino esta vacía se adiciona el elemento al final
        {
            listaS[i].AddUltimo(x);
            C[i] = C[i] + 1;        // incrementa en uno el contador de elemento de la lista i
        }
    }
    return listaS[i];             // retorna la lista i
}

//Funciona retorna TRUE si la lista i esta llena
public bool Llena(int i)
{
    return (C[i]==B[i]);
}

//Funcion retorna TRUE si la lista i esta vacia
public bool Vacía(int i)
{
    return (C[i]==0);
}

// funcion que retorna la cantidad de espacio disponible en la lista i
public int Dispos(int i)
{
    return (B[i]-C[i]);
}

// funcion que retorna la cantidad de elementos existentes en las N listas
public int Long()
{
    int s = 0;
    for (int i = 0; i < n; i++)
    {
        s = s + (B[i]-Dispos(i));
    }
    return s;
}
```

Pregunta 1

Correcta

Se puntúa 100,00 sobre 100,00

Se requiere gestionar n en listas enlazadas simples, considerar que cada lista tiene un tamaño máximo de elementos. La especificación formal del TAD Listas es la siguiente:

NOMBRE: TAD Listas**CONJUNTO:** N número natural ($0..n$), L_i es la identificación de la i -ésima lista enlazada ($0 \leq i < n$), n cantidad de listas enlazadas a gestionar**FUNCIONES**

- 1. **Create Listas()** // crea las n listas enlazadas
- 1. **AddS(L_i, x)** --> L_i // Adiciona el elemento x a la lista L_i
- 1. **LlenaS(L_i)** --> Boolean // Retorna true si L_i está llena
- 1. **DispoS(L_i)** -- > N // Retorna la cantidad de espacio disponible de la lista L_i
- 1. **Long()** --> N // Retorna la cantidad de elementos de las n listas

AXIOMAS**AddS(L_i, x)** ::= if **LlenaS(L_i)** then "ERROR"

Rellene el espacio en blanco con la lógica correcta para que funcionen correctamente las funciones de la clase clsListas.

NOTA: Todas las instrucciones debe ser rellena sin espacios en blanco, Si alguna función no funciona después de rellena las instrucciones, se descuenta el 40% de la nota final

```
// Clase del TAD ListaS
public class clsListaS
{
    //ATRIBUTOS DE LA CLASE
    private static int N = 5;           // Numero de Listas enlazadas simple a gestionar
    public clsLista[] listaS;           // Arreglo donde se almacena las N listas enlazadas
    public int[] B;                     // Arreglo donde se define el tamaño de cada lista i, para 0<=i<N
    public int[] C;                     // Arreglo para el contador de elementos de cada lista i, para 0<=i<N
    //CONSTRUCTOR DE LA CLASE
    public clsListaS()
    {
        listaS = new clsLista[N];
        B = new int[N];
        C = new int[N];
        for (int i = 0; i<N; i++)
        {
            listaS[i] = new clsLista(); // Se inicializa cada Lista i
            B[i] = ((i+1)*2);           // Se define el tamaño de cada Lista i
            C[i] = 0;                   // El contador de elementos de cada Lista i se coloca en cero
        }
    }
    // MODIFICA EL CONSTRUCTOR DE LA CLASE -- con el fin de inicializar los ATRIBUTOS

    //PROPIEDADES DE DESCRIPCION DE ACCESOS -- (get/set) que permite leer o cambiar los campos desde otras clases
    public clsLista[] Listas
    {
        get { return listaS; }
        set { listaS = value; }
    }
    //FUNCIONES DE LA CLASE
    //Funcion para adicionar un elemento x a la Lista i
    public clsLista AddS(int i, int x)
    {
        if (!LLena(i))                // Se adicionan elementos si no esta llena
            if (Vacía(i))              // Si esta vacía la lista i se adiciona el primer elemento
            {
                listaS[i].AddPrimero(x);
                C[i] = C[i] + 1;        // incrementa en uno el contador de elemento de la lista i
            }
            else                      // Sino esta vacía se adiciona el elemento al final
            {
                listaS[i].AddUltimo(x);
                C[i] = C[i] + 1;        // incrementa en uno el contador de elemento de la lista i
            }
        return
        listaS[i];
    }
    // retorna la lista i
    }
    //Funciona retorna TRUE si la lista i esta llena
    public bool LLena(int i)
    {
        return
        C[i]==B[i];
    }
    //Funcion retorna TRUE si la lista i esta vacía
    public bool Vacía(int i)
    {
        return
        C[i]==0;
    }
    // funcion que retorna la cantidad de espacio disponible en la lista i
    public int DispoS(int i)
    {
        return
        B[i]-C[i];
    }
}
```

```
✓
    }
    // funcion que retorna la cantidad de elementos existentes en las N listas
    public int Long()
    {
        int s = 0;
        for (int i = 0; i < n; i++)
        {
            s = s +
C[i];
        }
        return s;
    }
}
```

◀ Lab3-T7

Ir a...

[Resumen de retención de datos](#)

[Descargar la app para dispositivos móviles](#)