
	Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 1

## INFORME DE LABORATORIO

INFORMACION BASICA					
ASIGNATURA:	Tecnología de Objetos				
TITULO DE LA PRACTICA:	Scala, Programación funcional y orientada a objetos				
NUMERO DE LA PRACTICA:	01	AÑO LECTIVO:	2025 - B	N° SEMESTRE:	VI
FECHA DE PRESENTACION:	13 / 09 / 2025	HORA DE PRESENTACION:	--:-- PM		
INTEGRANTE (s): ▪ Huayhua Hillpa, Yourdyy Yossimar				NOTA:	
DOCENTE (s): ▪ Mg. Escobedo Quispe, Richart Smith					

### 1. Tarea

#### 1.1. Objetivo

- Conocer el lenguaje de programación Scala.
- Resolver un ejercicio de programación propuesto con Scala.
- Responder las preguntas para reforzar el aprendizaje.

#### 1.2. Desarrollo

- Visite el sitio oficial de Scala u otros para estudiar el lenguaje de programación Scala.
- Estudie el problema recursivo propuesto.
- Resuelva el problema con el lenguaje de programación Scala (Recursivamente).
- Responda las tres preguntas formuladas.



#### 1.3. Problema propuesto: Vuelto. Contando el vuelto o cambio.

Escriba una función recursiva que cuente de cuántas maneras diferentes puede dar cambio para una determinada cantidad y de acuerdo a una lista de denominaciones de monedas.  
Por ejemplo, hay 3 maneras de dar cambio si la cantidad=4 si tienes monedas con denominación 1 y 2:

1+1+1+1,

1+1+2,

2+2.

	Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 2

Realiza este ejercicio implementando la función `countChange` en Scala. Esta función toma una cantidad a cambiar y una lista de denominaciones únicas para las monedas.

Su definición es la siguiente:

```
def countChange(money: Int, coins: List[Int]): Int
```

Puedes usar las funciones `isEmpty`, `head` y `tail` en la lista de monedas enteras.

## 2. Equipos, materiales y temas utilizados

- Subsistema de Windows para Linux (WSL) con Ubuntu (versión predeterminada instalada mediante Microsoft Store).
- Sistema operativo: Microsoft Windows [Versión 10.0.26100.6584]
- MiKTeX-pdfTeX 4.15 (MiKTeX 23.4) L<sup>A</sup>T<sub>E</sub>X.
- Strawberry Perl (requerido por MiKTeX para la ejecución de scripts auxiliares en la compilación de ciertos paquetes).
- Helix 25.01.1 (e7ac2fcd)
- Visual Studio Code 1.104.0 x64
- Git version 2.41.0.windows.1
- Cuenta activa en GitHub para la gestión de repositorios remotos.
- Recursividad.
- Lenguaje de programación Scala.
- Compilador en línea [programiz.com](https://programiz.com).

## 3. URL de Repositorio Github



- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/yhuayhuahi/Teo.git>
- URL para el laboratorio (01) en el Repositorio GitHub.
- <https://github.com/yhuayhuahi/Teo/tree/main/laboratorios/lab01>

## 4. Desarrollo de las actividades

### 4.1. Actividades

#### 4.1.1. Función propuesta

Para resolver el problema propuesto se plantea la siguiente función recursiva en Scala:

	Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 3

Listing 1: Función countChange en Scala

```

1  def countChange(money: Int, coins: List[Int]): Int = {
2      if (money == 0) 1           // caso base hay 1 forma de dar cambio
3      else if (money < 0) 0       // si la cantidad es negativa
4      else if (coins.isEmpty) 0   // si no hay monedas disponibles no se puede dar cambio
5      else {
6          countChange(money - coins.head, coins) + countChange(money, coins.tail)
7      }
8  }

```

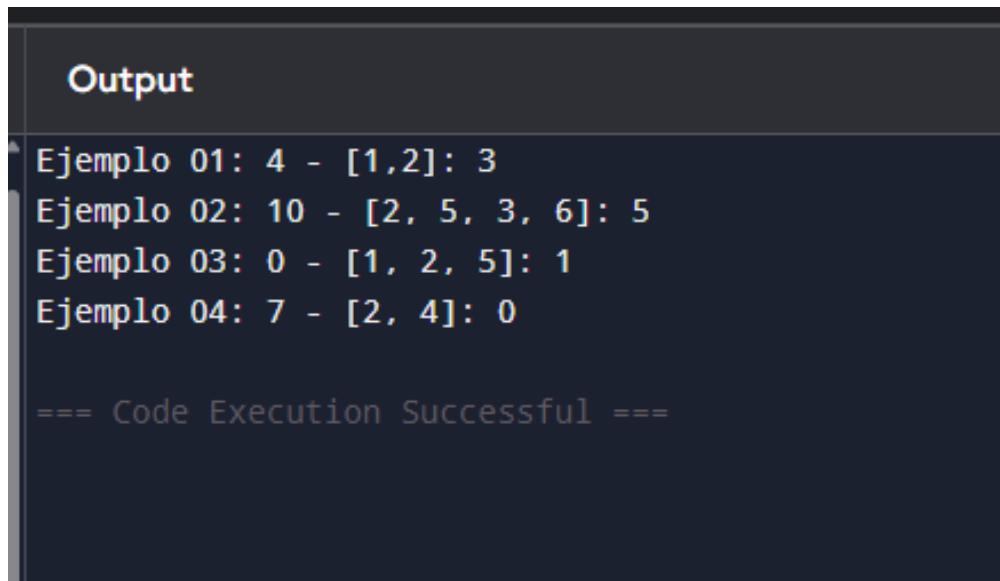
Se prueba el funcionamiento de la función en un compilador en línea, este es programiz.com:  
**Función main:**

Listing 2: Función main para probar el funcionamiento de countChange

```

1  @main def runCountChange(): Unit = {
2      println("Ejemplo 01: 4 - [1,2]: " + countChange(4, List(1, 2)))
3
4      println("Ejemplo 02: 10 - [2, 5, 3, 6]: " + countChange(10, List(2, 5, 3, 6)))
5
6      println("Ejemplo 03: 0 - [1, 2, 5]: " + countChange(0, List(1, 2, 5)))
7
8      println("Ejemplo 04: 7 - [2, 4]: " + countChange(7, List(2, 4)))
9  }

```



```

Output

Ejemplo 01: 4 - [1,2]: 3
Ejemplo 02: 10 - [2, 5, 3, 6]: 5
Ejemplo 03: 0 - [1, 2, 5]: 1
Ejemplo 04: 7 - [2, 4]: 0



=== Code Execution Successful ===

```

Figura 1: Ejecución de la función countChange en compilador en línea programiz.com

## 4.2. Estructura del laboratorio

A continuación se muestra la estructura de archivos y carpetas del laboratorio realizado:

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLD-001</p>	<p>Página: 4</p>

```

~/.../Teo/laboratorios/lab01  O  ¿main ≡ ?1 ~2
22:14:23 tree /f
Listado de rutas de carpetas para el volumen Windows
El número de serie del volumen es A025-E3A0
C:.
  README.md
  vuelto.scala
  informe-latex
    desarrollo.tex
    main.pdf
    main.tex
    rubricas.tex
  build
    main.aux
    main.fdb_latexmk
    main.fls
    main.log
    main.out
    main.pdf
    main.synctex.gz
  img
    logo_abet.png
    logo_episunsa.png
    prueba_ejecucion.png

```

Figura 2: Estructura de archivos y carpetas del laboratorio realizado

## 5. Cuestionario

### 5.1. Explique el caso base implementado.

En la función `countChange`, los casos base son los que permiten que la recursión se detenga. Son:

Listing 3: Casos base en la función `countChange`

```



1  if (money == 0) 1
2  else if (money < 0) 0
3  else if (coins.isEmpty) 0

```

- money == 0 → devuelve 1** Esto significa que si logramos llegar exactamente a 0, encontramos una forma válida de dar cambio (ya sea usando monedas o no).
- money < 0 → devuelve 0** Si el valor del dinero se vuelve negativo, significa que usamos más monedas de las necesarias. Ese camino no genera solución válida.
- coins.isEmpty → devuelve 0** Si ya no quedan monedas disponibles y todavía no llegamos a `money==0`, no podemos formar la cantidad. Entonces, no hay ninguna forma en ese camino.

### 5.2. ¿Este problema se resuelve mejor en el modelo de programación funcional?

Sí, este tipo de problema encaja muy bien en el modelo funcional por las siguientes razones:

	Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 5

1. El problema se define de manera recursiva: “¿De cuántas formas puedo dar cambio con las monedas actuales?” se puede dividir en dos subproblemas más pequeños (usar o no usar una moneda). La recursión es un pilar de la programación funcional.
2. En el enfoque funcional, las estructuras de datos (listas) son inmutables. Aquí no modificamos la lista de monedas ni el valor de money, simplemente pasamos nuevos valores a la recursión. Eso hace que el código sea más seguro y fácil de razonar.
3. En vez de indicar cómo construir las combinaciones paso a paso (enfoque imperativo con bucles y mutaciones), describimos el problema con reglas simples. El compilador y la recursión se encargan del “cómo”.

### 5.3. ¿Qué beneficios se podrían obtener para este problema utilizando memoización?

El algoritmo recursivo puro tiene un gran inconveniente el cual es que repite muchos cálculos.

Se podría optimizar el rendimiento utilizando memoización, que es una técnica para almacenar los resultados de subproblemas ya resueltos y reutilizarlos cuando se vuelven a necesitar. Los beneficios serían:



1. **Reducción de cálculos redundantes:** Almacenar los resultados de subproblemas evita que se recalculen múltiples veces, lo que mejora la eficiencia.
2. **Mejora en el tiempo de ejecución:** Con menos cálculos que realizar, el tiempo de ejecución del algoritmo se reduce significativamente.
3. **Facilidad para manejar problemas más grandes:** La memoización permite abordar problemas más complejos y de mayor tamaño sin un aumento exponencial en el tiempo de cálculo.

## 6. Rúbricas

### 6.1. Entregable Informe

Cuadro 1: Tipo de Informe

Informe	
Latex	El informe está en formato PDF desde Latex, con un formato limpio (buena presentación) y fácil de leer.

	Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas	
<b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
<b>Aprobación:</b> 2022/03/01	<b>Código:</b> GUIA-PRLD-001	<b>Página:</b> 6

## 6.2. Rúbrica para el contenido del Informe y demostración



- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple con el ítem correspondiente.
- Si un alumno supera la fecha de entrega, su calificación será sobre la nota mínima aprobada, siempre y cuando cumpla con todos los ítems.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Cuadro 2: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
2.0	0.5	1.0	1.5	2.0
4.0	1.0	2.0	3.0	4.0

Cuadro 3: Rúbrica para contenido del Informe y demostración

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4			
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	2	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	2	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	3	
<b>Total</b>		20		15	

	<p>Universidad Nacional de San Agustín de Arequipa Facultad de Ingeniería de Producción y Servicios Escuela Profesional de Ingeniería de Sistemas</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLD-001</p>	<p><b>Página:</b> 7</p>

## 7. Referencias

- [1] GeeksforGeeks, “What is Memoization? A Complete Tutorial”, GeeksforGeeks. URL: <https://www.geeksforgeeks.org/dsa/what-is-memoization-a-complete-tutorial/>
- [2] GeeksforGeeks, “Coin Change Problem in C++”, GeeksforGeeks. URL: <https://www.geeksforgeeks.org/cpp/coin-change-problem-in-cpp/>