


Evaluación Extraordinaria: Tiempo 3 horas	17 de Junio de 2020 - 16:00
Nombre y Apellidos	DNI/NIE: Firma:
Ciclo: CFGS Desarrollo de Aplicaciones Web Módulo: Programación . Turno: Vespertino	 IES Alonso de Avellaneda (Alcalá de Henares)

Esta prueba tiene la limitación de una nota máxima de 7

Si el/los ejercicio/s resultan plagio de recursos de Internet o de otras pruebas supondrá la inmediata calificación de 0 en la prueba mediante el proceso descrito en la modificación de la programación didáctica.

Rúbrica:

- Cuestión correcta en funcionamiento, buen formato, claridad, concisión, rigurosidad, precisión 100%.
- Incorrecto, incompleto, muy deficiente, poca claridad, ilegible, mal formato, plagio internet u otro recurso no permitido 0%.
- Si tiene algunas deficiencias que no impiden el funcionamiento, especificación, etc correctos, se tendrá en cuenta hasta un 50%.

Ejercicio 1: (2.5p): En una empresa financiera tiene una serie de códigos para realizar transacciones bancarias. Estos códigos se envían para que el destinatario compruebe dicho código de verificación en referencia a lo siguiente:

- Son códigos de 15 a 18 dígitos enteros positivos: 431345632242325, 769231219328129292
- Los números más significativos (más a la izquierda) tienen un valor, es un código de tipo de operación tal que:
 - 2: Envío de dinero: **envio**
 - 3: Petición de dinero: **peticion**
 - 43: Pago con tarjeta **tarjeta**
 - 76: Pago con transferencia: **transfer**
- Además, para verificar el código debe de realizarse la siguiente comprobación: todos los dígitos de la posición 0, 2, 4, 5, 8, etc. (pares) deben multiplicarse por 2 y sumarse por un lado siempre quedando una sola cifra. Es decir, si tenemos un dígito que será 8 multiplicamos por 2 saldría 16, cuyo resultado final será 7. Así: $5 \times 2 = 10 \Rightarrow 1 + 0 = 1$; $3 \times 2 = 6$. Así sucesivamente hasta que se obtiene toda la serie: $5 \times 2 = 10 \Rightarrow 1$, $3 \times 2 = 6 \Rightarrow 6$, $4 \times 2 = 8$, $2 \times 2 = 4$, $6 \times 2 = 12 \Rightarrow 3$, $4 \times 2 = 8$, $1 \times 2 = 2$, $4 \times 2 = 8$. Resultante $1+6+8+4+3+8+2+8 = 40$. Luego se suma el resto de dígitos: $2+2+2+3+5+3+3 = 40$. Se suman ambos y se comprueba que sea divisible por 10. Si no lo es, sería inválido.
- **El código de la transacción es válido sólo si cumple las 3 condiciones anteriores.**
- Implementar un programa que verifique si el código es válido y el tipo de operación.

431345632242325

Código de transferencia válido

Tipo operación: Pago con tarjeta

441345632242325

Código inválido

43134563224232

Código inválido

Ejercicio 2: (2.5p): Crea un programa que dando una longitud n , calcule la matriz $n \times n$ a partir del array dado. Calcula el resto de diagonales secundarias superiores en función de las siguientes condiciones:

- Se recogerá por teclado la dimensión y el vector `unaDim[n]` que será la diagonal principal.
- Se obtendrá la matriz `dosDim[n][n]` donde cada elemento de las diagonales secundarias superiores será `dosDim[i][j-1]+unDim[j]`
- Se imprimirá la matriz resultado de esta forma:

```
Introduce la longitud : 5
Introduce elemento en 0 : 1
Introduce elemento en 1 : 2
Introduce elemento en 2 : 3
Introduce elemento en 3 : 4
Introduce elemento en 4 : 5

  1    3    6   10   15
  0    2    5    9   14
  0    0    3    7   12
  0    0    0    4    9
  0    0    0    0    5
```

Ejercicio 3: (2.5p) Dada una lista de **caracteres** y un string de entrada, encontrar la ventana mínima en la que ese `string` contenga todos los caracteres. Por ejemplo: **entrada = ABBACBAA, caracteres = AAB** la ventana mínima será **BAA** porque son el mínimo conjunto de caracteres que tiene el `string` y que contiene a los caracteres (AAB) solicitados.

Se comienza al principio del array y nos movemos hacia la derecha. La ventana se obtendrá cuando se tengan los elementos requeridos. En el ejemplo se obtiene una ventana inicial de 4 elementos porque en este caso se han encontrado los elementos en esa ventana. Cuando se desliza a la siguiente, el último elemento de la ventana anterior pasa a ser el primero de la siguiente que también tiene otra ventana de 4 elementos. En el caso de que la última ventana tenga menos elementos que los caracteres buscados, se toma una ventana del tamaño que tenga los caracteres a buscar. Finalmente el programa mostrará la ventana mínima que contenga los caracteres solicitados. **Nota: no se pueden utilizar expresiones regulares ni Colecciones, etc, Sólo arrays, String, StringBuilder, etc.**

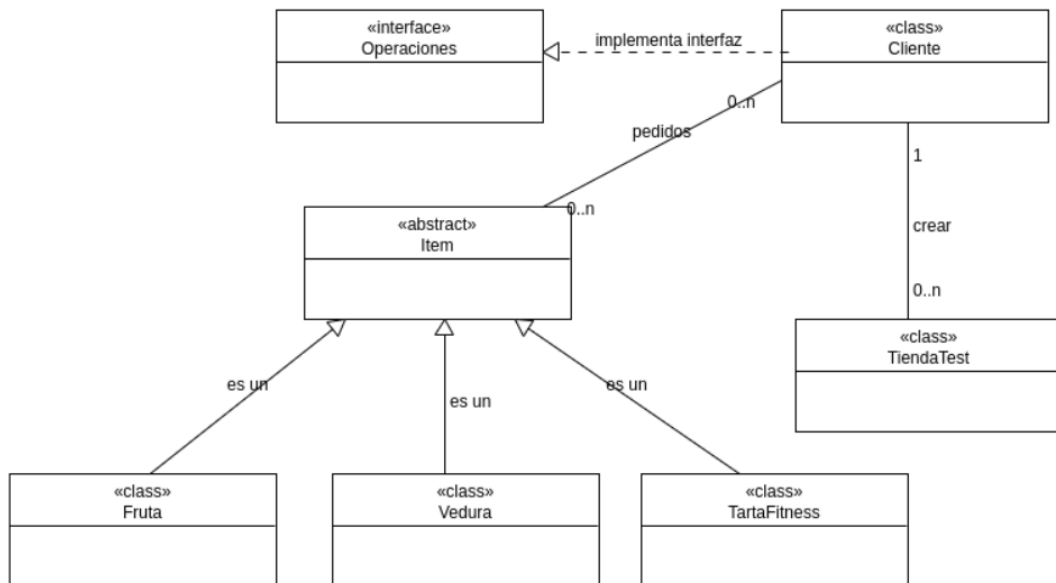
The diagram illustrates the sliding window technique for finding the longest substring without repeating characters. It shows three steps of the process:

- Step 1:** The initial window is `A B B A`. The character `B` at index 2 is highlighted with a red box, indicating a duplicate.
- Step 2:** The window is shifted to `B B A C`. The character `B` at index 3 is highlighted with a red box, indicating a duplicate.
- Step 3:** The window is shifted to `B A C B`. The character `B` at index 6 is highlighted with a red box, indicating a duplicate.

```
# Entrada
Introduce ventana:
ABBACBAA
Introduce chars:
AAB
# Salida
BAA
# Otro ejemplo
Introduce ventana:
ABCAAAAADBCAADFGHHHH
Introduce chars:
DAC
ADBC
```

Ejercicio 4: (2.5p) Se tiene una tienda donde se venden frutas, vegetales y *tartas fitness* vía pedidos online. Los clientes pueden hacer pedidos en cualquier cantidad de productos. Las frutas son pedidas mediante número de las mismas (p.e. 3 manzanas) y la verdura se vende al peso en kg (2 kg de patatas). Las *tartas fitness* se piden como items individuales y que llevan un mensaje en una tarjeta. Para hacer más eficiente los pedidos no se realizan estos sino se tiene al menos 10€ de cantidad total. Si están entre 10€ y 30€ se cargarán 7€ de gastos de envío. Y si los pedidos superan los 30€ el envío será gratis. Se pide realizar la jerarquía de clases con el funcionamiento correcto de la aplicación que describa el sistema anterior de pedidos y además una clase test que verifique de la siguiente forma según la siguiente salida:

[illegible]

[illegible]

Interfaz Operaciones: `void realizar()`: este método realiza la operación y calcula si el precio es < 10€ emite un mensaje que no se puede hacer el pedido si no tiene un pedido mínimo 10€. Añade coste 7€ de 10 a 30€ de pedido y más de 30€ sería gratis el envío.

Clase Cliente:

- atributos: nombre, apellido, direccion, cliente, ENVIO, pedido (lista)
- métodos:
 - Cliente(String, String, String): inicia atributos.
 - addItemAlPedido(Item): añade item al pedido
 - borrarItemDelPedido (int): borra el elemento n del pedido
 - mostrarDetallesDelPedido(): muestra nº producto, tipo, descripción, coste **de todos los productos del pedido.**
 - resumenPedido(): descripción y coste del pedido.

- `totalCostePedido()`: función que calcula el coste total de todos los productos del pedido sin los gastos de envío.
- Clase `Item`: clase abstracta con los siguientes métodos:
 - `calcularCoste()`: calcula el coste total del pedido
 - `describirltem()`: retorna la descripción del producto
 - `getTipo()`: retorna el tipo de producto: fruta, verdura o tartafitness.
- Clase `Vegetal`:
 - atributos: cantidad, descripción, preciokg, tipo
 - métodos: `Vegetal(double, double, String)`, `calcularCoste()`, `describirltem()`, `getTipo()`.
- Clase `Fruta`:
 - atributos: cantidad, descripción, preciokg, tipo
 - métodos: `Fruta(double, int, String)`, `calcularCoste()`, `describirltem()`, `getTipo()`