Boletín apoyo Java II

Ejercicio 1. Realiza un programa que calcule la nota que hace falta sacar en el segundo examen de la asignatura Programación para obtener la media deseada. Hay que tener en cuenta que la nota del primer examen cuenta el 40% y la del segundo examen un 60%.

Ejercicio 2. Realiza un programa que pinte una pirámide por pantalla. La altura se debe pedir por teclado. El carácter con el que se pinta la pirámide también se debe pedir por teclado.

Ejercicio 3. Muestra por pantalla todos los números primos entre 2 y 100, ambos incluidos.

Ejercicio 4. Muestra 50 números enteros aleatorios entre 100 y 199 (ambos incluidos) separados por espacios. Muestra también el máximo, el mínimo y la media de esos números.

Ejercicio 5. Escribe un programa que lea 10 números por teclado y que luego los muestre en orden inverso, es decir, el primero que se introduce es el último en mostrarse y viceversa.

Ejercicio 6. Escribe un programa que pida 10 números por teclado y que luego muestre los números introducidos junto con las palabras “máximo” y “mínimo” al lado del máximo y del mínimo respectivamente.

Ejercicio 7. Escribe un programa que pida 8 palabras y las almacene en un array. A continuación, las palabras correspondientes a colores se deben almacenar al comienzo y las que no son colores a continuación. Puedes utilizar tantos arrays auxiliares como quieras. Los colores que conoce el programa deben estar en otro array y son los siguientes: verde, rojo, azul, amarillo, naranja, rosa, negro, blanco y morado.

Ejercicio 8. En Italia un restaurante nos ha encargado una aplicación para colocar a los clientes en sus mesas. En una mesa se pueden sentar de 0 (mesa vacía) a 4 comensales (mesa llena). Cuando llega un cliente se le pregunta cuántos son. De momento el programa no está preparado para colocar a grupos mayores a 4, por tanto, si un cliente dice por ejemplo que son un grupo de 6, el programa dará el mensaje “Lo siento, no admitimos grupos de 6, haga grupos de 4 personas como máximo e intente de nuevo” . Para el grupo que llega, se busca siempre la primera mesa libre (con 0 personas). Si no quedan mesas libres, se busca donde haya un hueco para todo el grupo, por ejemplo si el grupo es de dos personas, se podrá colocar donde haya una o dos personas. Inicialmente, las mesas se cargan con valores aleatorios entre 0 y 4.

Cada vez que se sientan nuevos clientes se debe mostrar el estado de las mesas. Los grupos no se pueden romper aunque haya huecos sueltos suficientes. El funcionamiento del programa se ilustra a continuación.

Ejercicio 9. Crea una biblioteca de funciones matemáticas que contenga las siguientes funciones. Recuerda que puedes usar unas dentro de otras si es necesario. Observa bien lo que hace cada función ya que, si las implementas en el orden adecuado, te puedes ahorrar mucho trabajo. Por ejemplo, la función esCapicua resulta trivial teniendo voltea y la función siguientePrimo también es muy fácil de implementar teniendo esPrimo.

1. **esCapicua**: Devuelve verdadero si el número que se pasa como parámetro es capicúa y falso en caso contrario.
2. **esPrimo**: Devuelve verdadero si el número que se pasa como parámetro es primo y falso en caso contrario.
3. **siguientePrimo**: Devuelve el menor primo que es mayor al número que se pasa como parámetro.
4. **potencia**: Dada una base y un exponente devuelve la potencia.

5. **digitos**: Cuenta el número de dígitos de un número entero.
6. **digitoN**: Devuelve el dígito que está en la posición n de un número entero. Se empieza contando por el 0 y de izquierda a derecha.
7. **posicionDeDigito**: Da la posición de la primera ocurrencia de un dígitodentro de un número entero. Si no se encuentra, devuelve -1.
8. **quitaPorDetras**: Le quita a un número n dígitos por detrás (por la derecha).
9. **quitaPorDelante**: Le quita a un número n dígitos por delante (por la izquierda).
10. **pegaPorDetras**: Añade un dígito a un número por detrás.
11. **pegaPorDelante**: Añade un dígito a un número por delante.
12. **trozoDeNumero**: Toma como parámetros las posiciones inicial y final dentro de un número y devuelve el trozo correspondiente.
13. **juntaNumeros**: Pega dos números para formar uno.

Ejercicio 10. Realiza un buscador de sinónimos. Utiliza 20 palabras del diccionario español-inglés para el programa. El programa preguntará una palabra y dará una lista de sinónimos (palabras que tienen el mismo significado). Por ejemplo, si se introduce la palabra "frio", el programa dará como resultado: helado, fresco, viruji. ¿Cómo sabe el programa cuáles son los sinónimos de "frio"? Muy fácil, en el diccionario debe existir la entrada ("frio", "cold"), por tanto solo tendrá que buscar las palabras en español que también signifiquen "cold"; esta información estará en las entradas ("fresco", "cold") y ("Biruji", "cold"). Cuando una palabra existe en el diccionario pero no tiene sinónimos, debe mostrar el mensaje "No conozco sinónimos de esa palabra". Si una palabra no está en el diccionario se mostrará el mensaje "No conozco esa palabra". El usuario sale del programa escribiendo la palabra "salir".

Ejercicio 11. Realiza La asociación "Amigos de los pokemon" nos ha encargado una aplicación educativa sobre estos personajes. Crea un programa que pida al usuario el tipo de pokemon y que, a continuación, nos muestre su descripción y su habilidad. Si el tipo de anfibio introducido no existe, se debe mostrar el mensaje "Ese tipo de anfibio no existe".

Ejercicio 12. En ajedrez, el valor de las piezas se mide en peones. Una dama vale 9 peones, una torre 5 peones, un alfil 3, un caballo 2 y un péon vale, lógicamente, 1 peón. Realiza un programa que genere al azar las capturas que ha hecho un jugador durante una partida. El número de capturas será un valor aleatorio entre 0 y 15. Hay que tener en cuenta que cada jugador tiene la posibilidad de capturar algunas de las siguientes piezas (no más): 1 dama, 2 torres, 2 alfiles, 2 caballos y 8 peones. Al final debe aparecer la puntuación total