Algoritmos y Estructuras de Datos
Examen Final 04/02/2021 – Regulares 2015+ [Python y PE]

Un centro de alto rendimiento deportivo necesita un programa que le permita operar con los diferentes planes de entrenamiento que tiene disponibles. De cada **Plan**, se tiene una clave de identificación (una cadena que puede tener dígitos y caracteres), el nombre o título del plan (una cadena), la descripción del contenido detallado del plan (una cadena con un texto terminado en punto y con palabras separadas por un blanco (por ejemplo: "Plan semanal de caminata intensiva con al menos una hora de recorrido y seis kilómetros de alcance."), el monto a abonar para tomar ese plan, un número entre 1 y 10 que indica el tipo de plan (por ejemplo: 1: Básico, 2: Mantenimiento, 3: Deporte profesional, etc.), y otro número, pero entre 1 y 5 para indicar el rango de edad recomendable para tomar el plan (por ejemplo: 1: niños y adolescentes entre 6 y 14 años, 2: Jóvenes entre 15 y 18, etc.).

En base a lo anterior, desarrollar un programa completo que disponga al menos de dos módulos:

- En uno de ellos, definir la clase Plan que represente al registro a usar en el programa, y las funciones básicas para operar con registros de ese tipo.
- En otro módulo, incluir el programa principal y las funciones generales que sean necesarias. Para la carga de datos, aplique las validaciones que considere necesarias. El programa debe basarse en un menú de opciones para desarrollar las siguientes tareas:

1. Generar un **archivo binario** de registros que contenga los datos de todos los planes disponibles. El archivo debe generarse directamente sin crear primero un arreglo con los registros pedidos. Puede generarlo cargando los datos en forma manual o generando los datos en forma aleatoria. No se requiere que el archivo permanezca ordenado durante la carga. Debe considerar que esta opción puede ser invocada varias veces a lo largo del programa, y que en cada ejecución pueden agregarse al final del archivo tantos registros que desee el operador, sin eliminar los datos que ya estaban cargados.
2. Mostrar el archivo generado en el punto anterior, a razón de un registro por línea en la consola de salida.
3. A partir del archivo, generar un arreglo unidimensional de registros que contenga los datos de todos los planes cuyo monto a abonar sea mayor a un valor **p** que se carga por teclado. No se requiere que este arreglo se mantenga ordenado mientras se genera.
4. Mostrar todos los datos del arreglo que generó en el punto 3, pero de forma que el listado salga ordenado por clave de identificación.
5. Determine si existe en el arreglo un plan en el que el nombre del plan coincida con el valor **nom** que se carga por teclado. Si existe, retorne la cadena contenida en el campo **descripción** y detenga la búsqueda. Si no existe, retorne una cadena de la forma **'No existe'**. En ambos casos, muestre la cadena retornada.
6. Determine si existe en el arreglo un plan en el que la clave de identificación coincida con el valor **k** que se carga por teclado. Si existe, muestre sus datos completos y detenga la búsqueda. Si no existe, informe con un mensaje.
7. Determine el cuántos planes existen para cada una de las posibles combinaciones entre tipos de planes y rangos de edades (un total de 10 * 5 = 50 contadores). Muestre sólo los resultados que sean diferentes a cero.
8. Tome la cadena retornada en el punto 5, y determine cuántas palabras de esa cadena contenían cuatro o más caracteres de largo (de cualquier tipo) y al menos una vez la letra "t" y la letra "b" pero de forma que la "t" aparezca antes que la "b" (no es necesario que ambas letras estén consecutivas). Puede considerar que la cadena termina siempre con un punto, y que las palabras se separan entre ellas con un (y solo un) espacio en blanco. La cadena debe ser procesada caracter a caracter, a razón de uno por cada vuelta del ciclo que itere sobre ella.

Criterios generales de evaluación:

a.) Desarrollo del programa completo, incluyendo los dos módulos pedidos como mínimo, el menú correctamente planteado, funciones bien diseñadas y parametrizadas (cuando sea apropiado), validaciones cuando sean aplicables y estilo de código fuente: **[máximo: 2 puntos (8.3% del puntaje)]**

b.) Desarrollo correcto de los ítems 1 y 2: **[máximo: 4 puntos (16.6% del puntaje)]**

c.) Desarrollo correcto de los ítems 3 y 4: **[máximo: 4 puntos (16.7% del puntaje)]**

d.) Desarrollo correcto del ítem 5: **[máximo: 4 puntos (16.7 % del puntaje)]**

e.) Desarrollo correcto del ítem 6: **[máximo: 4 puntos (16.7% del puntaje)]**

f.) Desarrollo correcto del ítems 7: **[máximo: 2 puntos (8,3 % del puntaje)]**

g.) Desarrollo correcto del ítem 8: **[máximo: 4 puntos (16.7% del puntaje)]**

Para aprobar el parcial, el alumno debe llegar a un *total acumulado de al menos 60% del puntaje (es decir, alrededor de 14.4 puntos acumulados)*, pero obligatoriamente debe estar desarrollado el programa, funcionando y operativo.

NOTA	PORCENTAJE	CALIFICACIÓN
1		Insuficiente
2		Insuficiente
3		Insuficiente
4		Insuficiente
5		Insuficiente
6	60% a 68%	Aprobado
7	69% a 77%	Bueno
8	78% a 86%	Muy Bueno
9	87% a 95%	Distinguido
10	96% a 100%	Sobresaliente