Algoritmos y Estructuras de Datos II

Parcial 04-05: TEMA “A”

Ejercicio 1: Implementación de Tablas de Productos de panadería

Un empresario dueño de una cadena de panaderías decidió recolectar información de distintas sucursales de panaderías a lo largo del país, teniendo en cuenta temporada baja y alta. Para tal fin elaboró un archivo con información de cada uno de los productos más vendidos en cada sucursal, teniendo en cuenta tres ingredientes principales: harina (flour), Levadura (yeast) y Manteca (butter). El empresario notó que en cada sucursal usaban distinta cantidad de cada ingrediente para los productos, por diferencias de precios de estos en cada región.

Por este motivo, el empresario tomó nota de las cantidades de cada ingrediente, el precio y el valor de mercado de los productos terminados para cada sucursal en temporada alta y baja.

En el directorio del ejercicio se encuentran los siguientes archivos:

Archivo	***Descripción***
`main.c`	Contiene la función principal del programa
`bakery_product.h`	Declaraciones relativas a la estructura de los productos de panadería y de funciones de carga y escritura de datos.
`bakery_product.c`	Implementaciones **incompletas** de las funciones
`array_helpers.h`	Declaraciones / prototipos de las funciones que manejan la tabla de productos
`array_helpers.c`	Implementaciones **incompletas** de las funciones que manejan el arreglo

Abrir el archivo **`input/example_92.in`** para ver cómo se estructuran los datos.

Cada línea contiene los datos de la elaboración del mismo producto de panadería en distintas sucursales y si la temporada es alta o baja. Los dos primeros valores son el código de ciudad y la temporada (0 baja, 1 alta). Luego, para cada par ciudad-temporada tenemos las cantidades (en gramos), el precio (unitario, por cada gramo) de la harina, la levadura, la manteca y el valor de mercado del producto elaborado.

Lo que quiere decir que cada fila tiene el siguiente formato:

```
##<codigo_ciudad>??<temporada> (<cantidad_harina>, <precio_harina>)
(<cantidad_levadura>, <precio_levadura>) (<cantidad_manteca>,
<precio_manteca>) <valor_de_mercado>
```

Ejemplo:

```
##0??0 (200,10)  (5,2) (30,15) 3720
```

Consideraciones:

- Todas las cantidades son enteras, positivas
- Todos los precios son enteros positivos
- El valor de mercado es un entero positivo
- Los códigos de ciudades van del 0 al 4 (5 ciudades)
- La temporada es un entero sin signo con valores que pueden ser 0 o 1 (Low/baja, high/alta)
- No se podrá repetir la secuencia ciudad - temporada. Es decir, no puede existir dos entradas distintas para una combinación de ciudad y temporada.

El ejercicio consiste en completar el procedimiento de carga de datos en los archivos **`array_helpers.c`** y **`bakery_product.c`**. Recordar que el programa tiene que ser ***robusto***, es decir, debe tener un comportamiento bien definido para los casos en que la entrada no tenga el formato esperado.

Una vez completada la lectura de datos se puede verificar si la carga funciona compilando,

```
$ gcc -Wall -Werror -Wextra -pedantic -std=c99 -c array_helpers.c bakery_product.c main.c
$ gcc -Wall -Werror -Wextra -pedantic -std=c99 array_helpers.o bakery_product.o main.o -o
ptable
```

y luego ejecutar

```
$ ./ptable input/example_92.in
```

Ejercicio 2: Análisis de los datos

Completar la siguiente función definida en **`array_helpers:`**

```
unsigned int best_profit(BakeryProductTable ptable);
```

Esta función debe retornar la *mayor ganancia* que obtuvo una tienda en algún periodo

La ganancia se calcula restándole el costo del producto al valor de venta

Ejemplo:

```
##0??0 (200,10)  (5,2) (30,15) 3720
```

Para la ciudad `0` en temporada `baja` el costo de producir fue de

`(200 * 10) + (5 * 2) + (30 * 15) = 2460`

Mientras que el valor de mercado es de `3720` por lo tanto la ganancia fue de `3720 - 2460 = 1260`

Finalmente modificar el archivo **`main.c`** para que se muestre la mayor ganancia de la tabla

Nota: cada archivo de ejemplo incluye el resultado correcto en el nombre, es decir, para el ejemplo **`input/example_92.in`**, el resultado correcto es **92**.