UNIVERSIDAD DE MÁLAGA
Dpto. Lenguajes y Ciencias de la Computación
E.T.S.I. Informática

Fundamentos de la Programación
Examen 1ª Convocatoria Ordinaria **31/01/20**

Apellidos, Nombre:

Titulación:

Grupo:

Código PC usado:

NOTAS PARA LA REALIZACIÓN DEL EXAMEN:

- La solución se almacenará en la carpeta **EXAMENFEBFP**, dentro de **Documentos**. Si la carpeta ya existe, debe borrarse todo su contenido. En otro caso, debe crearse.
- Los nombres de los ficheros con la solución para los ejercicios 1, 2, 3 y 4 serán **ejercicio1.cpp**, **ejercicio2.cpp, ejercicio3.cpp** y **ejercicio4.cpp,** respectivamente.
- Al inicio del contenido de cada fichero deberá aparecer un comentario con **el nombre del alumno, titulación, grupo y código del equipo** que se está utilizando (cada dato en una línea diferente)**.**
- **Debe consultarse** el documento "**Obligaciones y Recomendaciones Estilo de Programación**", disponible en el Campus Virtual de la asignatura, con objeto de tener en cuenta los puntos allí señalados en las soluciones a los ejercicios.
- Una vez terminado el examen, se subirán los ficheros ***.cpp** a la tarea creada en el **campus virtual** para ello.
- **No está permitido**:
  - Utilizar documentación electrónica o impresa.
  - Intercambiar documentación con otros compañeros.

(**1 pto) 1.- Diseña** un **algoritmo** que lea de teclado una secuencia de longitud arbitraria de números enteros positivos separados por un espacio y terminada en 0, y muestre por pantalla el mayor de los números primos leídos. Un número primo es un número entero positivo mayor que 1 que es divisible únicamente por él mismo y por la unidad. En caso de que no haya ningún primo en la secuencia de entrada, se mostrará un mensaje indicándolo. Puedes suponer que la entrada es correcta.

Importante: La puntuación de este problema será de 1 punto sólo en el caso de que el algoritmo funcione correctamente. En otro caso, la puntuación será de 0 puntos.

Dos ejemplos de ejecución:

```
Introduzca una secuencia de enteros positivos acabada
en 0: 1 3 5 7 2 9 17 13 18 0

El mayor primo de la secuencia es: 17
```

```
Introduzca una secuencia de enteros positivos acabada
en 0: 1 9 15 8 12 0

No hay ningun primo en la secuencia
```

(**2 ptos) 2.-** Una matriz de números enteros *doblemente estocástica normalizada* es aquella que cumple las siguientes condiciones:

1. Todos sus elementos son no negativos y menores que 100.
2. La suma de los elementos de cada fila es igual a 100.
3. La suma de los elementos de cada columna es igual a 100.

Diseña un **algoritmo** que lea de teclado una colección de números enteros con los que rellenar una matriz cuadrada de tamaño NxN (siendo N una constante definida), muestre por pantalla después el contenido de dicha matriz y finalmente indique si dicha matriz es o no doblemente estocástica normalizada. Recuerda que en la solución se valorará la eficiencia de la misma y el uso de diseño descendente.

Dos ejemplos de ejecución (N = 3):

20	30	50
50	0	50
30	70	0

```
Introduce los numeros enteros para una matriz cuadrada de 3x3:
20 30 50
50 0 50
30 70 0

La matriz introducida es:
20 30 50
50 0 50
30 70 0

La matriz introducida SI es doblemente estocastica normalizada
```

20	30	50
55	-5	50
25	75	0

```
Introduce los numeros enteros para una matriz cuadrada de 3x3:
20 30 50
55 -5 50
25 75 0

La matriz introducida es:
20 30 50
55 -5 50
25 75 0

La matriz introducida NO es doblemente estocastica normalizada
```

(**4 ptos) 3.-** **Diseña** un **algoritmo** para gestionar las cuentas de gastos comunes realizadas por un grupo de amigos.

Para ello, el algoritmo deberá leer de teclado los gastos comunes realizados por cada persona (el nombre, como una cadena de caracteres sin espacios en blanco, y la cantidad de dinero gastada, la cual es un número real) hasta que se introduzca como nombre `FIN`, considerando que no habrá más de `MAX = 10` personas distintas. En caso de que la misma persona realice múltiples gastos comunes, entonces se unificarán los gastos comunes de esa persona en un único gasto común con la suma de las cantidades gastadas. Una vez leídos todos los datos, se mostrarán por pantalla los nombres de las personas introducidas junto con sus gastos totales realizados (ver ejemplo de ejecución).

A continuación, calculará cuánto dinero debe pagar o recibir cada persona y a quién se lo debe pagar o de quién lo debe recibir, para que al final los gastos de todas las personas sean iguales. Para ello, realizará los siguientes cálculos:

1. Calcula la media de los gastos comunes totales (total de gastos dividido entre el número de personas distintas), mostrando por pantalla el resultado (ver ejemplo de ejecución).
2. Calcula cuánto debe pagar o recibir cada persona, según su total de gastos y la media calculada anteriormente. Muestra por pantalla el mensaje correspondiente (ver ejemplo de ejecución).
3. Repite el siguiente proceso hasta que todas las personas tengan los mismos gastos comunes, es decir, que todos los gastos sean aproximadamente igual a la media (cada gasto debe ser mayor o igual que *`media-0.01`* y menor o igual que *`media+0.01`*). Dicho de otra forma, hasta que las cantidades que deben pagar o recibir todas las personas sean aproximadamente cero (valor absoluto menor que *0.01*):
    a. Selecciona la persona P_1 que debe pagar más dinero (aquella que tiene el menor gasto) y la persona P_2 que debe recibir más dinero (aquella que tiene el mayor gasto).
    b. Se calcula cuánto dinero debe pagar P_1 a P_2, como la menor de las dos cantidades anteriores.
    c. Se muestra el mensaje de cuánto dinero paga una persona a la otra (ver ejemplo de ejecución).
    d. Se transfiere el dinero de una persona a la otra, ajustando las cantidades que deben pagar o recibir.

Ejemplo de ejecución:

Entrada:

```
Introduzca nombres y gastos (FIN para terminar)
 Nombre: pepe
 Gastos: 20
 Nombre: lola
 Gastos: 30
 Nombre: pepe
 Gastos: 10
 Nombre: juan
 Gastos: 40
 Nombre: lola
 Gastos: 20
 Nombre: luis
 Gastos: 20
 Nombre: ana
 Gastos: 30
 Nombre: eva
 Gastos: 34
 Nombre: FIN
```

Salida:

```
pepe ha gastado en comun 30
lola ha gastado en comun 50
juan ha gastado en comun 40
luis ha gastado en comun 20
ana ha gastado en comun 30
eva ha gastado en comun 34

La media de gastos en comun es 34

pepe debe pagar 4
lola debe recibir 16
juan debe recibir 6
luis debe pagar 14
ana debe pagar 4
eva esta a la par

luis paga 14 a lola
pepe paga 4 a juan
ana paga 2 a lola
ana paga 2 a juan
```

(**3 ptos) 4.-** Se considera que una palabra P1 es menor que otra P2 si la suma del código ASCII de las letras que forman P1 es menor que la suma del código ASCII de las letras que forman P2. **Diseña** un **algoritmo** que lea de teclado una secuencia indefinida de palabras terminada en `FIN` y muestre por pantalla aquellas palabras que sean menores (según la definición proporcionada anteriormente) que la primera palabra leída en la secuencia. Además, en la salida **las palabras se mostrarán ordenadas** de menor a mayor según su orden alfabético y **sin repeticiones**.

Ejemplo de ejecución:

Entrada:

```
Introduzca un texto (FIN para terminar): CREO QUE VOY A IR ESTA
TARDE AL CINE Y LUEGO VOY A IR A CENAR MAS TARDE FIN
```

Salida:

```
Las palabras que son menores que CREO son:

A AL CINE IR MAS QUE VOY Y
```

NOTAS:

- El texto contiene un número indefinido de palabras.
- El texto termina con la palabra FIN.
- Cada palabra tiene un número indefinido pero limitado de caracteres (todos alfabéticos mayúsculas).
- En el texto habrá un número máximo MAX_PAL_DIST (una constante) de palabras distintas.
- El carácter separador de palabras es el espacio en blanco.