Apellidos, Nombre:_____ DNI: _____

INSTRUCCIONES

Realiza los ejercicios tal como se detallan en cada uno de ellos. Se proporcionan varios folios vacíos para poder anotar lo que se necesite de la resolución del examen, el contenido de esos folios no se evaluará, pero serán entregados al profesor.

Descargue el zip proporcionado en Moodle llamado "*AlternativaBloqueII-ApellidosNombre.zip*". Renombre la carpeta sustituyendo "*ApellidosNombre*" por sus dos apellidos y nombre. Por ejemplo, si mi nombre es Antonio Manuel Durán Rosal, el directorio pasa de "*AlternativaBloqueII-ApellidosNombre.zip*" a llamarse "*AlternativaBloqueII-DuranRosalAntonioManuel.zip*" (recuerde que no puede haber **ni espacios ni caracteres especiales**, como la ñ, ni tildes).

El proyecto consta de los siguientes archivos:

En el directorio *src*:

- ✓ ***ejercicio1/*** directorio que contiene lo necesario para la realización del ejercicio 1
- ✓ ***ejercicio2/*** directorio que contiene lo necesario para la realización del ejercicio 2

El examen puede ser resuelto utilizando cualquier editor de textos y su correspondiente compilación y ejecución en el sistema operativo Linux. **Al profesor sólo se le subirán los códigos fuente sin incluir el ejecutable.** Aunque a continuación se detallan los ejercicios y el qué hay que realizar, a lo largo de los ficheros aparece la palabra **TODO** que indicará lo mismo.

Hay que entregar:

- ➢ **En Moodle el directorio completo en formato .zip con el nombre "*AlternativaBloqueII-ApellidosNombre.zip*" (poner los apellidos y nombre del alumno).**
- ➢ **Al profesor, el examen con la resolución de las preguntas teóricas 1.b) y 3.**

Antes de darle a **enviar** los documentos en Moodle, **llamar al profesor** para comprobar que todo es correcto. **No se aceptarán envíos sin la supervisión del profesor**.

Confirmo que he leído las instrucciones del examen.

Firma y hora de entrega:

Apellidos, Nombre:_____ DNI: _____

EJERCICIO 1 **Tiempo Estimado: 40minutos** **Puntuación: 4.25 puntos**

En un concurso de cocina, un número de cocineros (potencia de dos) participan para ganar el título del "Cocinero del Año". El concurso consiste en que los participantes tienen un minuto para escoger un número de ingredientes que pueden utilizar en sus recetas y luego deben colocarse en una fila. Los participantes se llevarán el premio si la fila cumple con la siguiente propiedad:

- La suma de los ingredientes de los cocineros en la primera mitad de la fila debe ser igual a la suma de los ingredientes de los cocineros en la segunda mitad de la fila. Además, esta condición debe cumplirse de forma recursiva en cada mitad de la fila.

Por ejemplo:

- Marta, 2 ingredientes
- Luis, 4 ingredientes
- Ana, 1 ingredientes
- Pedro, 5 ingredientes
- Lucía, 3 ingredientes
- Javier, 3 ingredientes
- Daniel, 4 ingredientes
- Sofía, 2 ingredientes

Para verificar si una fila es válida, se utilizará una función llamada **validarFilaIngredientes** que devolverá la suma total de los ingredientes si se cumple la condición o -1 en caso contrario. Cuando la fila tiene dos cocineros, siempre se cumple la condición y se devuelve el número de ingredientes de ambos cocineros.

Se pide:

a) Defina la clase Cocinero que tendrá dos atributos: nombre y numero_ingredientes. La clase debe tener un constructor con estos dos parámetros, métodos observadores y modificadores, sobrecarga del operador =, y la función de salida mostrar para mostrar de forma compacta → (Nombre, numero_ingredientes). No hace falta incluir ninguna otra función. **(0.75 puntos)**

b) Explica la estrategia que vas a utilizar para la resolución del ejercicio. **(0.5 puntos)**

c) Implemente la función **validarFilaIngredientes** que devuelva la suma de los ingredientes si se cumple la propiedad, o -1 en caso contrario. **(3 puntos)**

El programa principal probará con las siguientes 5 filas, de las cuales las dos primeras cumplirán las condiciones y devolverán el número total de ingredientes, mientras que las restantes devolverán -1.

Se probarán las siguientes aulas:

Marta, 2 – Luis, 4 – Ana, 1 – Pedro, 5 – Lucía, 3 – Javier, 3 – Daniel, 4 – Sofía, 2

Marta, 3 – Luis, 6 – Ana, 8 – Pedro, 1 – Lucía, 5 – Javier, 4 – Daniel, 2 – Sofía, 7

Marta, 3 – Luis, 6 – Ana, 6 – Pedro, 3 – Lucía, 4 – Javier, 1 – Daniel, 2 – Sofía, 3

Marta, 3 – Luis, 6 – Ana, 3 – Pedro, 6 – Lucía, 4 – Javier, 2 – Daniel, 4 – Sofía, 8

Marta, 3 – Luis, 7 – Ana, 2 – Pedro, 1 – Lucía, 5 – Javier, 2 – Daniel, 1 – Sofía, 24

Apellidos, Nombre:_____ DNI: _____

La salida del programa debería ser así:

```
Primera fila:
(Marta, 2) (Luis, 4) (Ana, 1) (Pedro, 5) (Lucía, 3) (Javier, 3) (Daniel, 4) (Sofía, 2)

Segunda fila:
(Marta, 3) (Luis, 6) (Ana, 8) (Pedro, 1) (Lucía, 5) (Javier, 4) (Daniel, 2) (Sofía, 7)

Tercera fila:
(Marta, 3) (Luis, 6) (Ana, 6) (Pedro, 3) (Lucía, 4) (Javier, 1) (Daniel, 2) (Sofía, 3)

Cuarta fila:
(Marta, 3) (Luis, 6) (Ana, 3) (Pedro, 6) (Lucía, 4) (Javier, 2) (Daniel, 4) (Sofía, 8)

Quinta fila:
(Marta, 3) (Luis, 7) (Ana, 2) (Pedro, 1) (Lucía, 5) (Javier, 2) (Daniel, 1) (Sofía, 24)

El resultado de la primera fila es: 24
El resultado de la segunda fila es: 36
El resultado de la tercera fila es: -1
El resultado de la cuarta fila es: -1
El resultado de la quinta fila es: -1
```

Notas:

- Podrá incluir los métodos que necesite en los archivos *cocinero.h* y *cocinero.cpp*, pero se evaluará el funcionamiento de los métodos que se piden.

- El programa principal *main.cpp* no se tendrá que cambiar, sólo comentar y descomentar cuando sea necesario.

- Sólo se permitirá la definición de funciones en el archivo .h, cuando éstas se traten de **constructores, observadores y modificadores**. En cualquier otro caso, serán declaradas en el fichero .h y definidas en su correspondiente fichero .cpp.

- Usar el calificador *const.*

- Usar las directivas #ifndef #define #endif.

- No modificar nada de los archivos main, solo descomentar los comentarios que sean necesarios.

- Comprobar los TODO.

- Ante cualquier duda, contactar con el profesor.

Apellidos, Nombre:_____ DNI: _____

EJERCICIO 2 **Tiempo Estimado: 50minutos** **Puntuación: 6 puntos**

Dado un grupo de clientes C = {c1, c2, …, cn), y una serie de centros de distribución D = {d1, d2, …, dm}, cada centro dj tiene capacidad para abastecer ciertos clientes, y un costo de distribución pj. El problema consiste en seleccionar un subgrupo de centros de distribución con el menor costo total, de forma que todos los clientes sean abastecidos por al menos un centro.

Para ello tenéis disponible un esqueleto formado por:

- **centro.h** y **centro.cpp**: representa la clase con los clientes que abastece el centro y el costo de distribución.

- **solucion.h** y **solucion.cpp**: almacena los centros de distribución seleccionados, el conjunto de los clientes abastecidos y el costo total. Nota: se usa una lista de conjuntos auxiliar para acelerar los cálculos.

- **estado.h** y **estado.cpp**: representa un estado para usar en los algoritmos **backtracking y voraz**.

- **problema.h** y **problema.cpp**: contiene lo necesario para ejecutar los algoritmos **backtracking y voraz**.

Se pide:

centro.h y centro.cpp

a) Sobrecarga del operador < **(0.25 puntos)**

solucion.h y solucion.cpp

b) Sobrecarga del operador > **(0.25 puntos)**

estado.h y estado.cpp

c) Implemente la función avanza **(0.25 puntos)**

d) Implemente la función retrocede **(0.25 puntos)**

e) Implemente la función getAlternativas **(1 punto)**

f) Implemente la función getMejorAlternativa **(1 punto)**

g) Implemente la función esFinal (**0.25 puntos)**

problema.h y problema.cpp

h) Implemente la función ejecutaBacktracking **(0.25 puntos)**

i) Implemente la función bt **(1 punto)**

j) Implemente la función ejecutaVoraz **(0.25 puntos)**

k) Implemente la función vorazR **(0.5 punto)**

l) Implemente la función actualizaMejorSolucion **(0.75 puntos)**

Apellidos, Nombre:_____ DNI: _____

La salida del programa debería ser así:

```
Voraz:
Centros:
{Ana Antonio Pedro (30)}
{Ana Mercedes (25)}
{Alfonso Ana Mercedes (40)}
Clientes abastecidos: { Alfonso Ana Antonio Mercedes Pedro }
Costo total: 95

Backtracking:
Centros:
{Antonio Pedro (22)}
{Alfonso Ana Mercedes (40)}
Clientes abastecidos: { Alfonso Ana Antonio Mercedes Pedro }
Costo total: 62
```

Notas:

- Podrá incluir los métodos que necesite en los archivos proporcionados, pero se evaluará el funcionamiento de los métodos que se piden.

- El programa principal *main.cpp* no se tendrá que cambiar, sólo comentar y descomentar cuando sea necesario.

Apellidos, Nombre:_____ DNI: _____

EJERCICIO 3 **Tiempo Estimado: 5minutos** **Puntuación: 0.75 puntos**

Explique brevemente el problema que resolviste en la actividad de Backtracking realizada por grupos, incluyendo: problema a resolver, planteamiento, peculiaridades y problemas que tuvisteis durante la realización del mismo, así como la solución para resolverlos.