Nombre: .....
Apellidos: .....
DNI: .....

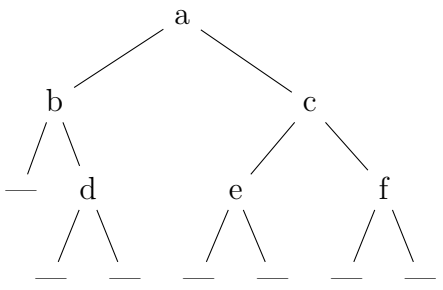
Estructuras de Datos y Algoritmos — Examen final extraordinario 2019/20
Grado en Desarrollo de Videojuegos. 2º V
Facultad de Informática, UCM

Instrucciones:
Esta primera parte del examen dura **50 minutos** y tiene una puntuación total de **3pt**.
No se puede usar el ordenador, móvil, calculadora o cualquier otro dispositivo electrónico.

Recordatorio

$\sum_{i=a}^{n}(s_i \pm t_i) = \sum_{i=a}^{n} s_i \pm \sum_{i=a}^{n} t_i$	$\sum_{i=a}^{n} k \cdot s_i = k \cdot \sum_{i=a}^{n} s_i$	$\sum_{i=a}^{n} i = \frac{(a+n)(n-a+1)}{2}$	$\sum_{i=0}^{n-1} k^i = \frac{1-k^n}{1-k}$
$(a+b)^2 = a^2 + b^2 + 2ab$	$(a-b)^2 = a^2 + b^2 - 2ab$		

1. (0.35 pt) Explica cómo se calcula el recorrido **inorden** de un árbol binario. Indica cuál sería el recorrido inorden del siguiente árbol binario.



2. (0.5 pt) Calcula el coste asintótico de las siguientes dos funciones, indicando y explicando los principales pasos realizados.

```
1  int f1(int n) {
2    int r = 1;
3    while (n > 1) {
4      r = r * n;
5      n = n - 1;
6    }
7    return r;
8  }
```

```
9
10 int f2(int m) {
11   int r = 0;
12   for (int i = 0; i < m; ++i) {
13     r = r + f1(i);
14   }
15   return r;
16 }
```

3. (0.4 pt) Indica qué coste tiene el siguiente fragmento de código, teniendo en cuenta que `l` es una variable de tipo `list <int>` usando la **implementación vista en clase**. ¿Cómo se podría mejorar?

```
1          int acc = 0;
2          for (unsigned int i = 0; i < l.size(); ++i)
3            acc += l.at(i);
```

4. (0.4 pt) Obtén la recurrencia de coste $T(n)$ de la siguiente función recursiva. **No se pide expandir la recurrencia ni obtener el coste asintótico.**

```
1  int frec(int n) {
2    if (n < 10) {
3      return 2*n;
4    } else {
5      int acc = 0;
6      int i = 1;
7      while (i <= n) {
8        acc += frec(n-1);
9        i += 1;
10     }
11     return acc;
12   }
13 }
```

5. (0.35 pt) Lista el conjunto mínimo de operaciones de un árbol binario de búsqueda (BST) e indica su coste asintótico en el caso peor considerando la representación de nodos enlazados vista en clase.

6. (1 pt) A partir de la siguiente recurrencia $T(n)$ que representa el coste de un algoritmo recursivo, utiliza el método de las expansiones para calcular en qué orden de complejidad está incluida $T(n)$. Se deben realizar **todos los pasos** e indicar claramente el resultado obtenido en cada uno de ellos.

$$T(n) = \begin{cases} 4 & \text{si } n < 2 \\ 2T(\frac{n}{2}) + 7 & \text{si } n \geq 2 \end{cases}$$

Estructuras de Datos y Algoritmos — Examen final ordinario 2019/20
Grado en Desarrollo de Videojuegos. 2º V
Facultad de Informática, UCM

Instrucciones:

- Esta segunda parte del examen dura **2 horas y 10 minutos**.
- En el enlace «**Material para descargar**» dentro del juez tenéis un ZIP con las transparencias de la asignatura, el código de los TADs, las plantillas de solución del juez y otros documentos.
- En el enlace «**C++ reference**» dentro del juez (botón en la parte superior) tenéis disponible la documentación completa de la STL.
- Se valorará la calidad del código, la eficiencia, la inclusión del coste de las funciones y métodos involucrados y la corrección con respecto a los casos de prueba.
- Si no os da tiempo a terminar el ejercicio incluid un comentario lo más completo posible describiendo la idea de la solución y su coste. Aunque no haya nada de código, este comentario puede ser evaluable.
- El fichero enviado debe contener una cabecera indicando vuestro nombre completo, usuario del juez utilizado y puesto de laboratorio donde se ha realizado.

1. (4 pt) Un fábrica dispone de N máquinas y tiene N empleados contratados. Cada máquina está muy automatizada, así que únicamente necesita que un empleado se encargue de ella. Además, al inicio de cada día se negocia qué maquinas puede manejar cada empleado, además de fijar qué sueldo percibirá cada empleado al manejar cada máquina (puede ser diferente de una máquina a otra).

Al empezar cada día, el dueño de la fábrica debe enviar a los empleados a las distintas máquinas para comenzar la producción. Sin embargo, no le interesa cualquier asignación sino que quiere minimizar la suma de sueldos que debe pagar a sus empleados. Además, si detecta que con las restricciones impuestas no es posible asignar un empleado a cada máquina entonces establecerá el día como festivo y cerrará la fábrica.

Implementa una solución a este problema siguiendo una de las dos técnicas algorítmicas tratadas en la asignatura: *divide y vencerás* o *vuelta atrás*.

Entrada

La entrada consta de varios casos de prueba. Cada caso comienza con una línea indicando el número $0 < N \leq 15$ de máquinas y empleados. A continuación le siguen N líneas, cada una mostrando los sueldos de cada empleado i en orden creciente de 1 a N. Para cada empleado i, la entrada contiene una secuencia de N números separados por espacios «$S_{i1}\ S_{i2} \ldots S_{iN}$» donde $0 \leq S_{ij} \leq 100$ indica el sueldo que cobrará el empleado i al manejar la máquina j. Si algún $S_{ij} = 0$ entonces el trabajador i no puede manejar la máquina j ese día. La entrada termina con un caso especial con el valor de $N = 0$ que no debe procesarse.

Salida

Para cada caso de prueba la salida será una línea. Si no es posible encontrar alguna asignación de empleados a máquinas, la línea contendrá la palabra `FESTIVO`. En otro caso, mostrará la mínima suma de sueldos que el dueño tiene que pagar para asignar una máquina a cada

empleado.

Ejemplo de entrada

```
1
18
2
3 8
2 10
2
5 0
7 6
3
3 0 0
8 0 0
1 2 3
0
```

Ejemplo de salida

```
18
10
11
FESTIVO
```

Estructuras de Datos y Algoritmos — Examen final ordinario 2019/20
Grado en Desarrollo de Videojuegos. 2º V
Facultad de Informática, UCM

Instrucciones:

- Esta segunda parte del examen dura **2 horas y 10 minutos**.
- En el enlace «**Material para descargar**» dentro del juez tenéis un ZIP con las transparencias de la asignatura, el código de los TADs, las plantillas de solución del juez y otros documentos.
- En el enlace «**C++ reference**» dentro del juez (botón en la parte superior) tenéis disponible la documentación completa de la STL.
- Se valorará la calidad del código, la eficiencia, la inclusión del coste de las funciones y métodos involucrados y la corrección con respecto a los casos de prueba.
- Si no os da tiempo a terminar el ejercicio incluid un comentario lo más completo posible describiendo la idea de la solución y su coste. Aunque no haya nada de código, este comentario puede ser evaluable.
- El fichero enviado debe contener una cabecera indicando vuestro nombre completo, usuario del juez utilizado y puesto de laboratorio donde se ha realizado.

2. (3 pt) Cada vez que un cliente realiza un gasto con una tarjeta de crédito, en «*Flojencio Bank*» reciben un mensaje indicando el número de tarjeta y el importe del gasto. De la misma manera, cuando un cliente realiza una devolución el banco recibe un mensaje similar pero con una cantidad negativa representando el importe devuelto. Como los empleados de «*Flojencio Bank*» son muy perezosos, hasta que no llega fin de mes no se preocupan por calcular el gasto acumulado de cada tarjeta para cargarlo en la cuenta de los clientes. Ha llegado ese momento, y necesitan urgentemente que realices un programa para calcular el gasto acumulado de una serie de tarjetas de crédito.

Para solucionar este problema debes utilizar la **clase o clases de la STL** que mejor se adapten, teniendo en cuenta la adecuación y la eficiencia obtenida.

Entrada
La entrada consta de varios casos de prueba. Cada caso de prueba comienza con una línea con un número $M > 0$ que indica el número de gastos/reembolsos de ese mes. A continuación le siguen M líneas conteniendo un número de tarjeta de crédito TC (formado por 16 dígitos) seguido de un espacio y un número entero indicando la cantidad de euros del gasto o reembolso. Después aparece una línea conteniendo el número $C > 0$ de tarjetas de crédito para las que queremos conocer el gasto final. Por último, siguen C líneas, cada una conteniendo un número de tarjeta de crédito TC siguiendo el mismo formato descrito anteriormente.

La entrada termina con un caso especial con $M = 0$, que no debe procesarse.

Salida
Para cada una de las C tarjetas de crédito solicitadas en el caso de prueba, se debe mostrar una línea con el formato «TC G», donde G es el gasto acumulado para la tarjeta de crédito

TC, y entre el número de tarjeta de crédito TC y el gasto G hay **exactamente un espacio**. Si la tarjeta TC solicitada no aparecía en ningún gasto o reembolso del caso de prueba entonces su gasto será 0. Al final de cada caso de prueba debéis insertar una **línea en blanco**.

Ejemplo de entrada

```
3
4050623861094020 56
0073793578923534 12
4050623861094020 -10
3
4050623861094020
4873790098620712
0073793578923534
2
4873790836598465 12
4873790836598465 -13
1
4873790836598465
0
```

Ejemplo de salida

```
4050623861094020 46
4873790098620712 0
0073793578923534 12

4873790836598465 -1

```