Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Estructuras de Datos, 2022-10
Parcial 1 - 25 de febrero de 2022

Reglas y recomendaciones

Durante el parcial, deben observarse las siguientes reglas:

1. El parcial se debe desarrollar en el computador, escribiendo las respuestas en los campos designados dentro del cuestionario de BrightSpace.

2. Archivos adicionales se aceptarán sólo en el caso del diseño, de acuerdo a las instrucciones que se encuentran en el enunciado del parcial. Guarde regularmente su trabajo (para evitar posibles pérdidas de información), con los nombres de archivo
`par1_apellido1_apellido2_nombre1_nombre2.txt` (para el diseño de los TADs) y
`par1_apellido1_apellido2_nombre1_nombre2_dis.pdf` (para el diagrama de relación entre TADs)
Tenga en cuenta que los nombres y apellidos van todos en minúsculas, los espacios se reemplazan por guiones bajos y no se utilizan ni tildes ni 'ñ'. Estos archivos deben enviarse únicamente dentro de la pregunta correspondiente en el cuestionario de BrightSpace.

3. El parcial tiene una duración de dos horas, contadas a partir del inicio normal de la clase. El cuestionario en BrightSpace está programado para cerrarse al terminar las dos horas, por lo que no enviarlo dentro de ese lapso de tiempo se considerará como parcial no presentado y tendrá una calificación de 0.0.

4. El parcial es estrictamente individual y se debe desarrollar únicamente en la sala de computadores del día.

5. Se recomienda no utilizar compiladores, intérpretes de lenguajes de programación o entornos de desarrollo de cualquier tipo.

6. Puede utilizar sus apuntes, libros, e Internet para obtener la información que necesite para el parcial. Sin embargo, está absolutamente prohibido comunicarse con cualquier otro ser humano para obtener información sobre el parcial, a través de cualquier medio (conversación directa, MSN, Skype, gtalk, whatsapp, blackberry messenger, etcétera).

7. Los celulares deben permanecer apagados, y no se debe enviar ni recibir ningún mensaje de texto.

8. La única excepción a lo anterior son los profesores y monitores de la asignatura, quienes sólo responderán consultas respecto a la claridad de las preguntas del parcial y no responderán consultas sobre la materia.

9. Si el estudiante incumple con cualquiera de las reglas, será evaluado con nota 0.0

10. **Recuerde que diseñar no implica implementar.**

1 (15%) Análisis de código

Considere la siguiente función (se garantiza que los argumentos contienen la cantidad de elementos necesarios):

```
typedef std::vector< unsigned int > TVector;
typedef std::list< unsigned int > TList;

short func( TList l[7], TVector v ) {
  v.clear( );

  for( int i = 0; i < 7; i++ ) {

    TList::iterator lIt;
    unsigned int j = 0;
    for( lIt = l[i].begin( ); lIt != l[i].end(); lIt++ ) {
      if ( *lIt % 5 == 0 )   j++;
    }

    v.push_back( j );
  }

  short c = 0;
  for( int i = 0; i < v.size(); i++ ) {
    if( v[i] != 0 )   c++;
  }

  return c;
}
```

1.1 (7%) ¿Qué hace «func»? (descríbala en 12 palabras o menos)

1.2 (8%) ¿Cuál es el orden de complejidad de «func»? Justifique brevemente su respuesta.

2 (24%) Selección múltiple con única respuesta

2.1 (8%) Jorge tiene un conjunto de n libros ordenado alfabéticamente. Primero, toma uno a uno (en orden) los libros y los apila. Luego, saca uno a uno los libros de la pila, hasta dejar encima un libro cuyo título empieza por 'G'. ¿Cuáles son los libros que quedan fuera de la pila?

1. Los libros con títulos que comienzan entre la 'G' y la 'Z'.
2. Los libros con títulos que comienzan entre la 'A' y la 'G'.
3. Los libros con títulos que comienzan entre la 'H' y la 'Z'.
4. Los libros con títulos que comienzan entre la 'A' y la 'H'.

2.2 (8%) En una lista doblemente encadenada de la STL (`std::list< T >`), el iterador que permite recorrerla es de tipo:

1. de acceso aleatorio.
2. bidireccional.
3. una lista no tiene iterador.
4. de sólo lectura.

2.3 (8%) Para realizar un cálculo se cuenta con un algoritmo dividido en dos bloques secuenciales de instrucciones. El primer bloque de instrucciones tiene complejidad $O(n^2)$, mientras que el segundo bloque de instrucciones tiene complejidad $O(n \log n)$. ¿Cuál es la complejidad del algoritmo completo?

1. $O(n \log n^2)$
2. $O(n^3 \log n)$
3. $O(n^2)$
4. $O(n \log n)$

3 (61%) Diseño e Implementación de TADs

En la producción de útiles escolares la fábrica *MiPintor* es considerada la mejor del mercado. En particular, sus témperas son muy demandadas en las diferentes papelerías, misceláneas y tiendas de la ciudad. Su éxito radica, al parecer, en un proceso de producción de mucho cuidado y que ofrece gran variedad de colores en sus témperas, así como cajas con diversas combinaciones de colores.

Cada uno de los tarritos de témpera dentro de la fábrica tiene un código identificador de 3 caracteres, que usualmente son dos letras y un número (AM4, RO2, AZ6, VE1, ...) y describe el color de la misma. También se incluye una pequeña descripción textual de los componentes utilizados en la creación del color de témpera, junto con la proporción de cada uno; algo así como: "Caseína 30%, Agua 50%, Pigmento 20%". Finalmente, para los procesos de auditoría en la fabricación de las témperas, es necesario almacenar una cadena de caracteres con el día, mes y año de producción ("22/02/2022"), junto con el tiempo (en minutos) que demora la fabricación (o la mezcla de los elementos) para cada tarrito de témpera.

Una vez llenos y cerrados, los tarritos de témpera se almacenan en colas individuales por colores, es decir, hay una cola de tarritos de cada uno de los colores que se producen en la fábrica. Estas colas están numeradas e identificadas cada una con el código del color que almacena (AM4, RO2, ...). De allí serán tomados los tarritos para armar las cajas de colores combinados; y en este caso el orden es importante, para que puedan ir usándose los tarritos que tienen más tiempo de fabricación.

Al momento de armar una caja de témperas, se cuenta con una lista de los códigos de colores que se desean incluir en la caja. Normalmente, se pueden armar cajas de 6, 12 y 24 colores diferentes, y de acuerdo a la cantidad de colores la caja se identifica con un nombre diferente, con el que se conocerá el producto en el proceso de mercadeo y venta. De esta forma, con la lista de los colores requeridos, un operario se dirige a las colas de tarritos y va extrayendo un tarrito de cada cola para ponerlo en la caja respectiva. Si por alguna razón no se cuenta con alguno de los colores de la lista, la caja no puede ser completada, y el comprador deberá esperar a que se pueda producir el color requerido.

La fábrica *MiPintor* realiza hasta ahora estos procesos de forma manual, pero considera necesario implementarlos en un sistema de información, para facilitar la expansión de las ventas y de la compañía. Con los ingenieros de sistemas que se encargarán de este proceso, se han identificado inicialmente dos procedimientos importantes, que pueden ser desarrollados en el sistema de información:

1. Simular el proceso de armado de una caja de témperas, teniendo en cuenta el nombre que se le dará y la lista de los códigos de colores que debe contener. Inicialmente, se debe validar que la cantidad de colores corresponda a los tamaños de caja permitidos (6, 12 o 24), y que los códigos representen colores diferentes entre sí (una caja no debería tener tarritos de colores repetidos). De esta forma, es posible simular el proceso físico de armado de una caja, al poner uno a uno los tarritos extraídos de las colas de los colores dados en la caja respectiva.

2. Identificar el tiempo promedio requerido para la fabricación de cada uno de los colores disponibles dentro de la fábrica. Estos tiempos, dados en minutos, son importantes para el proceso de auditoría en la fabricación, pues permiten identificar demoras en la fabricación o malas prácticas en el mezclado de los componentes.

Se le pide entonces diseñar e implementar (en C++) los componentes ya descritos del sistema de información para la fábrica *MiPintor*.

3.1 (16%) Diseño

Diseñe el sistema y el (los) TAD(s) solicitado(s). Utilice la plantilla de especificación de TADs vista en clase para el diseño. Recuerde que diseñar es un proceso previo a la implementación, por lo que no debería contener ninguna referencia a lenguajes de programación (es decir, si escribe encabezados o código fuente, el punto no será evaluado y tendrá una calificación de cero). Para simplicidad del diseño, no es necesario incluir los métodos obtener y fijar (get/set) del estado de cada TAD.

3.2 (7%) Diagrama de relación

El diseño incluye el diagrama de relación entre TADs, cuando se definen dos o más TADs en el punto anterior. En ese caso, adjúntelo en formato PDF, JPG o PNG como parte de su entrega.

3.3 (21%) Operación 1: Armar una caja de témperas.

Dado el (los) TAD(s) ya diseñado(s), escriba la implementación en C++ del algoritmo que permite armar una caja de témperas de acuerdo a un nombre dado, un tamaño elegido y a la lista de códigos de colores de los tarritos de témpera que la caja debe contener. La implementación deberá tener en cuenta:

- la definición apropiada de los prototipos de los métodos/funciones (i.e. recibir/retornar los datos suficientes y necesarios para su correcta ejecución),
- el NO uso de salidas/entradas por pantalla/teclado (i.e. paso/retorno correcto de valores y/o objetos),
- el correcto uso del diseño definido en el punto anterior, y
- la escritura de todo el código que pueda llegar a necesitar que no esté incluido en la STL.

3.4 (17%) Operación 2: Calcular el tiempo promedio de fabricación por colores.

Dado el (los) TAD(s) ya diseñado(s), escriba la implementación en C++ del algoritmo que permite calcular el tiempo promedio de fabricación de un tarrito de témpera para cada uno de los colores disponibles en la fábrica. Así como en el punto anterior, la implementación deberá tener en cuenta:

- la definición apropiada de los prototipos de los métodos/funciones (i.e. recibir/retornar los datos suficientes y necesarios para su correcta ejecución),
- el NO uso de salidas/entradas por pantalla/teclado (i.e. paso/retorno correcto de valores y/o objetos),
- el correcto uso del diseño definido en el punto anterior, y
- la escritura de todo el código que pueda llegar a necesitar que no esté incluido en la STL.