Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Estructuras de Datos, 2023-30
Parcial 1 - 16 de agosto de 2023

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 (9:00a.m.). El cuestionario en BrightSpace está programado para cerrarse al terminar las dos horas (11:00a.m.), 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. Puede utilizar también herramientas de inteligencia artificial (tipo ChatGPT), bajo su propio riesgo y corroborando la información obtenida.

7. Sin embargo, está absolutamente prohibido comunicarse con cualquier otro ser humano para obtener información sobre el parcial, a través de cualquier medio (conversación directa, Teams, Skype, Zoom, Gtalk, Whatsapp, etcétera). También está totalmente prohibido copiar o transcribir texto o código de enunciados y respuestas de parciales anteriores, o de diseños e implementaciones que no sean propias.

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

9. 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.

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

11. **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 para la correcta ejecución de la función):

```
typedef std::vector< string > TVector;
typedef std::list< string >   TList;
typedef std::list< TList >    TList2;
void f ( TList2& l, TVector v[] ) {

  for ( TList2::iterator it = l.begin( ); it != l.end( ); it++ ) {
    for ( TList::iterator it2 = it->begin( ); it2 != it->end( ); it2++ ) {
      if ( (*it2) == string("si") ) {
        v[0].push_back(*it2);
      } else if ( (*it2) == string("no") ){
        v[1].push_back(*it2)
      }
    }
  }
}
```

1.1 (7%) ¿Qué hace «func»? (descríbala en 15 palabras o menos)

Llena dos listas con las repeticiones de “si” y “no” en la multilista.

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

O(n^2), hay dos ciclos anidados, el externo sobre la longitud de la multilista (variable) y los internos sobre la longitud de cada una de las sublistas (variable).

2 (21%) Selección múltiple con única respuesta

2.1 (7%) Juanito toma una secuencia de números enteros desde 1 hasta n e inserta la primera mitad en una cola y la segunda mitad en una pila. Luego, llena una lista extrayendo primero los elementos de la pila, y luego los elementos de la cola. ¿Cómo quedan los elementos en la lista final?

1. Los primeros $n/2$ en orden inverso, luego los segundos $n/2$ en orden inverso.
2. Los segundos $n/2$ en orden inverso, luego los primeros $n/2$ en orden.
3. Los primeros $n/2$ en orden inverso, luego los segundos $n/2$ en orden.
4. Los segundos $n/2$ en orden inverso, luego los primeros $n/2$ en orden inverso.

2.2 (7%) Para cualquier contenedor secuencial estándar de la STL, el iterador `reverse_iterator` permite:

1. Recorrido principal hacia atrás, sólo lectura.
2. Recorrido principal hacia adelante, lectura/escritura.
3. Recorrido principal hacia atrás, lectura/escritura.
4. Recorrido principal hacia adelante, sólo lectura.

2.3 (7%) 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 $O(\log_{10} n)$. ¿Cuál es la complejidad del algoritmo completo?

1. $O(n^2 \log_{10} n)$
2. $O(n^2)$
3. $O(\log_{10} n)$
4. $O(n^2 + \log_{10} n)$

3 (64%) Diseño e Implementación de TADs

Las conferencias académicas o científicas (también conocidas por otros nombres como congresos, simposios, talleres o reuniones) son espacios dispuestos para que investigadores, integrantes de la industria, académicos y hasta el público en general puedan presentar y discutir sus trabajos de investigación, profundización, o temas de actualidad. Son espacios importantes en el quehacer de docentes, investigadores y científicos, pues facilitan el intercambio de información entre investigadores, incentivando la práctica de habilidades de presentación efectiva, la retroalimentación y comentarios entre pares académicos, la posibilidad de concretar oportunidades conjuntas de trabajo y colaboración, y la posibilidad de estar actualizado en la investigación actual en el área de interés.

Estas conferencias se programan para uno o varios días de actividades secuenciales, donde cada día puede dedicarse a un subtema específico dentro de la temática general de la conferencia. Durante cada día, pueden realizarse algunas de las siguientes actividades, organizadas en bloques y siempre en forma secuencial: presentaciones cortas (`short`) en bloques de 1 hora para presentar trabajos específicos, presentaciones principales (`main`) de 1 hora y media realizadas por expertos en el área, paneles de discusión (`panel`) alrededor de un tema entre varios expertos invitados, exposiciones de posters (`poster`) en las que trabajos específicos se presentan a través de un formato afiche, y talleres guiados (`workshop`) en los que los asistentes pueden profundizar en tecnologías o herramientas específicas. Actividades adicionales, como recesos (`pause`) y almuerzos o comidas (`lunch`) se intercalan entre éstas para poder dar a los asistentes tiempos de descanso. Cada una de estas actividades suele tener asociado un título o nombre, un tipo (uno entre `short`, `main`, `panel`, `poster`, `workshop`, `pause` o `lunch`) una hora de inicio y una duración estimada (en minutos). En el caso de las presentaciones cortas y las exposiciones de posters, en éstos se encolan por orden de llegada los trabajos aceptados para presentación en la conferencia, los cuales se identifican por un número asignado al momento del envío. A continuación se presenta un ejemplo de organización de actividades en una conferencia:

<table>
<tr><th></th><th>Martes 13 de junio</th><th>Miércoles 14 de junio</th><th>Jueves 15 de junio</th></tr>
<tr><td>8:00am</td><td rowspan="2"></td><td></td><td></td></tr>
<tr><td>8:30am</td><td rowspan="2">Panel de discusión
8:30am - 90 minutos</td><td rowspan="2">Taller 2
8:30am – 90 minutos</td></tr>
<tr><td>9:00am</td><td rowspan="2">Conferencia principal 1
9am – 60 minutos</td></tr>
<tr><td>9:30am</td><td>Receso
10am – 30 minutos</td><td>Receso
10am – 30 minutos</td><td>Receso
10am – 30 minutos</td></tr>
<tr><td>10:00am</td><td rowspan="3">Presentaciones cortas 1
10:30am – 90 minutos
Trabajos: 4, 7, 23, 45</td><td rowspan="3">Presentaciones cortas 2
10:30am – 90 minutos
Trabajos: 6, 15, 25, 36</td><td rowspan="3">Presentaciones cortas 4
10:30am – 90 minutos
Trabajos: 8, 12, 27, 40</td></tr>
<tr><td>10:30am</td></tr>
<tr><td>11:00am</td></tr>
<tr><td>11:30am</td><td rowspan="3">Almuerzo libre
12m – 120 minutos</td><td rowspan="3">Almuerzo libre
12m – 120 minutos</td><td rowspan="3">Almuerzo libre
12m – 120 minutos</td></tr>
<tr><td>12:00m</td></tr>
<tr><td>12:30pm</td></tr>
<tr><td>1:00pm</td><td rowspan="3">Conferencia principal 2
2pm – 60 minutos</td><td rowspan="4">Presentaciones cortas 3
2pm – 90 minutos
Trabajos: 9, 18, 21, 38</td><td rowspan="3">Conferencia principal 3
2pm – 60 minutos</td></tr>
<tr><td>1:30pm</td></tr>
<tr><td>2:00pm</td></tr>
<tr><td>2:30pm</td><td rowspan="2">Receso
3pm – 30 minutos</td><td rowspan="2">Receso
3pm – 30 minutos</td></tr>
<tr><td>3:00pm</td></tr>
<tr><td>3:30pm</td><td rowspan="3">Taller 1
3:30pm – 90 minutos</td><td>Receso
3:30pm – 30 minutos</td><td rowspan="2">Conferencia principal 4
3:30pm – 60 minutos</td></tr>
<tr><td>4:00pm</td><td rowspan="2">Exposición de pósteres
4pm – 60 minutos
Trabajos: 50, 53, 56, 59, 64, 67, 73, 78, 81, 85</td></tr>
<tr><td>4:30pm</td><td rowspan="2"></td></tr>
<tr><td>5:00pm</td><td></td><td></td></tr>
</table>

Para facilitar la programación de actividades en cualquier conferencia, se espera poder desarrollar una herramienta informática que organice por días los bloques de actividades y permita almacenar de ellos la información necesaria. Dentro de esta herramienta, se ha identificado inicialmente una operación muy importante: la que permite agregar una nuevo trabajo a una sesión de pósteres o de presentaciones cortas

de la conferencia. Esta operación debe recibir los siguientes datos: día, título de la actividad, hora de inicio y tipo de la actividad, y número identificador del trabajo a agregar. Con la información de la actividad verifica que la actividad sí esté programada en el día y hora específicos, y además que la actividad sea de tipo poster o short para garantizar que se puede encolar un nuevo trabajo. También es importante verificar que el trabajo que se quiere agregar no esté ya en la cola de trabajos de la actividad.

3.1 (22%) 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.

```
TAD Hora
Datos mínimos:
- h, entero, representa la hora.
- m, entero, representa los minutos.
Operaciones:
- ObtenerH(), retorna la hora actual.
- ObtenerM(), retorna los minutos actuales.
- FijarH(nh), cambia la hora a nh.
- FijarM(nm), cambia los minutos a nm.
- HorasIguales(hora2), compara la hora actual con otra para determinar si son iguales.

TAD Trabajo
Datos mínimos:
- id, entero, identificador del trabajo.
- aceptado, booleano, representa si el trabajo está aceptado en la conferencia.
Operaciones:
- ObtenerId(), retorna el identificador actual del trabajo.
- ObtenerAceptado(), retorna el estado de aceptación actual del trabajo.
- FijarId(nid), cambia el identificador a nid.
- FijarAceptado(nacept), cambia el estado de aceptación a nacept.

TAD Actividad
Datos mínimos:
- titulo, cadena de caracteres, representa el título o nombre de la actividad.
- tipo, cadena de caracteres, representa el tipo de actividad (short, main, panel,
  poster, workshop, pause o lunch).
- horaInicio, Hora, representa la hora de inicio de la actividad.
- duracion, entero, representa la duración en minutos de la actividad.
- trabajos, cola de Trabajo, almacena la información de los trabajos que se presentarán
  durante la actividad (sólo para tipo short y poster).
Operaciones:
- ObtenerTitulo(), retorna el título actual.
- ObtenerTipo(), retorna el tipo actual.
- ObtenerHora(), retorna la hora de inicio actual.
- ObtenerDuracion(), retorna la duración actual.
- ObtenerTrabajos(), retorna la cola de trabajos actual.
- FijarTitulo(ntitulo), cambia el título a ntitulo.
```

- FijarTipo(ntipo), cambia el tipo a ntipo.
- FijarHora(nhora), cambia la hora de inicio a nhora.
- FijarDuracion(ndur), cambia la duración a ndur.
- FijarTrabajos(ntrab), cambia la cola de trabajos a ntrab.
- AgregarTrabajo(nid), agrega un nuevo trabajo, identificado por nid, a la cola de trabajos de forma ordenada.

TAD Dia
Datos mínimos:
- tema, cadena de caracteres, representa el subtema específico de la conferencia a tratar en el día.
- actividades, lista de Actividad, almacena las actividades programadas para el día.

Operaciones:
- ObtenerTema(), retorna el tema actual.
- ObtenerActividades(), retorna la lista de actividades actual.
- FijarTema(ntema), cambia el tema actual a ntema.
- FijarActividades(nacts), cambia la lista de actividades a nacts.
- AgregarAActividad(titulo, tipo, inicio, idTrabajo), agrega un nuevo trabajo, identificado por idTrabajo, a la actividad identificada con el título, tipo y hora de inicio dados.

TAD Conferencia
Datos mínimos:
- programa, lista de Dia, representa la programación de actividades en los diferentes días de la conferencia.

Operaciones:
- ObtenerPrograma(), retorna la lista actual de días.
- FijarPrograma(nprog), cambia la lista de días a nprog.
- AgregarAActividad(dtema, titulo, tipo, inicio, idTrabajo), agrega un nuevo trabajo, identificado por idTrabajo, a la actividad identificada por titulo, tipo e inicio, dentro del día cuyo tema está dado por dtema.

3.2 (10%) 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 (16%) Diseño: Agregar trabajo a actividad

Dado el (los) TAD(s) ya diseñado(s), diseñe la operación que permite agregar un nuevo trabajo a una actividad de pósteres o conferencias cortas ya programada en la conferencia. El diseño deberá especificar, en lenguaje natural, los pasos necesarios para agregar un nuevo trabajo, teniendo en cuenta los datos y operaciones ya definidos en el (los) TAD(s).

Pasos a seguir:
1. Moverse en el programa de la conferencia (lista de Dia) hasta ubicarse en el día requerido.
2. Dentro del día, moverse en la lista de actividades, hasta ubicarse en la actividad requerida (validar título, tipo y hora de inicio).
3. Si se encuentra la actividad requerida, del tipo adecuado, revisar los trabajos encolados para verificar que no existe otro trabajo con el mismo identificador, y

encolar el nuevo trabajo en la posición adecuada.

4. Esta revisión implica desencolar todos los trabajos, o al menos hasta encontrar el deseado, o hasta encontrar que el número del trabajo podría quedar ordenado al ubicarlo allí, y luego volver a encolar todo lo que está por fuera para restituir la cola a su estado inicial.

3.4 (16%) Implementación: Agregar trabajo a actividad

Dado el (los) TAD(s) ya diseñado(s), y el diseño de la operación escrito en el punto anterior, escriba la implementación en C++ del algoritmo que permite agregar un nuevo trabajo a una actividad de pósteres o conferencias cortas ya programada en la conferencia. 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),
- la coherencia y el correcto uso del diseño definido en los puntos anteriores, y
- la escritura de todo el código que pueda llegar a necesitar que no esté incluido en la STL.

```
bool Conferencia::AgregarAActividad (string dtema, string titulo, string tipo,
                                     Hora inicio, int idTrabajo) {
  bool res = false;
  std::list<Dia>::iterator it;
  for (it = programa.begin(); it != programa.end(); it++)
    if (it->ObtenerTema() == dtema)
      res = it->AgregarActividad(titulo,tipo,inicio,idTrabajo);
  return res;
}

bool Dia::AgregarAActividad (string titulo, string tipo, Hora inicio, int idTrabajo) {
  bool res = false;
  std::list<Actividad>::iterator it;
  for (it = actividades.begin(); it != actividades.end(); it++)
    if (it->ObtenerTitulo() == titulo && it->ObtenerTipo() == tipo
        && it->ObtenerHora().HorasIguales(inicio))
      res = it->AgregarTrabajo(idTrabajo);
  return res;
}

bool Actividad::AgregarTrabajo(int idTrabajo) {
  bool res = false;
  bool dup = false;
  std::queue<Trabajo> aux;
  Trabajo elTrabajo;
  while (!trabajos.empty()) {
    elTrabajo = trabajos.front();
    trabajos.pop();
    if (elTrabajo.ObtenerId() == idTrabajo)
      dup = true;
```

```
    aux.push(elTrabajo);
  }
  if (dup) {
    FijarTrabajos(aux);
  } else {
    Trabajo nuevoT;
    nuevoT.FijarId(idTrabajo);
    nuevoT.FijarAceptado(true);
    while (!aux.empty()) {
      elTrabajo = aux.front();
      aux.pop();
      if (elTrabajo.ObtenerId() > idTrabajo)
        trabajos.push(nuevoT);
      trabajos.push(elTrabajo);
    }
    res = true;
  }
  return res;
}

bool Hora::HorasIguales(Hora hora2) {
  bool res = false;
  if (h == hora2.ObtenerH() && m == hora2.ObtenerM())
    res = true;
  return res;
}
```