INF-134 Estructuras de Datos, 2023-2

Tarea 2

Profesores: Gabriel Carmona, Javier Robledo, Carolina Saavedra
gabriel.carmonat@sansano.usm.cl, javier.robledo@usm.cl,carolina.saavedra@usm.cl

Ayudantes:
Juan Alegria (juan.alegria@usm.cl)
Tomás Barros (tomas.barros@sansano.usm.cl)
Diego Debarca (diego.debarca@usm.cl)
Rodrigo Flores (rodrigo.floresf@usm.cl)
Sebastián Torrealba (sebastian.torrealba@usm.cl)

Fecha de entrega: 31 de octubre, 2023.
Plazo máximo de entrega: 5 días.

1. Reglas del Juego

La presente tarea debe hacerse en grupos de 3 personas. Toda excepción a esta regla debe ser conversada con los ayudantes (usando los correos indicados en el encabezado de esta tarea). No se permiten de ninguna manera grupos de más de 3 personas. Debe usarse el lenguaje de programación C++. Al evaluarlas, las tareas serán compiladas usando el compilador `g++`, usando la línea de comando `g++ archivo.cpp -o output -Wall`. **No se aceptarán variantes o implementaciones particulares de g++, como el usado por MinGW**. Se deben seguir los tutoriales que están en Aula USM, cualquier alternativa explicada allí es válida. Recordar que una única tarea en el semestre puede tener nota menor a 30. El no cumplimiento de esta regla implica reprobar el curso. No se permite usar la biblioteca *stl*, así como ninguno de los *containers* y algoritmos definidos en ella (e.g. `vector`, `list`, etc.). Está permitido usar otras funciones de utilidad definidas en bibliotecas estándar, como por ejemplo `math.h`, `string`, `fstream`, `iostream`, etc.

2. Objetivos

Entender y familiarizarse con la implementación y utilización de estructuras de datos tipo listas.

Además se fomentará el uso de las buenas prácticas y el orden en la programación de los problemas correspondientes.

3. Problemas a Resolver

En esta sección se describen los problemas a resolver implementando programas, funciones o clases en C++. Se debe entregar el código para cada uno de los problemas en archivos `.cpp` separados (y correspondientes `.hpp` de ser necesario).

3.1. Problema 1: Descomposición de Bloques

Un profesor de la UTFSM estudió una estructura de datos **DeBlock**. Esta consiste en almacenar una secuencia de números utilizando lista de listas. Para lograr esto, esta estructura recibe la secuencia de números y la divide en bloques de tamaño b. Por ejemplo, si se tiene la secuencia $\{1, 3, 2, 5, 6, 2, 3\}$ y $b = 3$, esto generaría la lista de listas siguiente $\{\{1, 3, 2\}, \{5, 6, 2\}, \{3\}\}$. Ahora esta estructura debe permitir hacer las siguientes operaciones: `insert`, `length` y `get_value`.

El profesor no tiene tiempo para programarlo, entonces se le ocurrió hacerlo como tarea para los estudiantes de EDD.

3.1.1. Requisitos

Su programa deberá constar de un *TDA* llamado `DeBlock` que va almacenar elementos tipo `tElem`. El TDA tendrá el siguiente formato:

```
class DeBlock {
private:
    int cant_elems;
    tListadeLista l;
public:
    DeBlock(tElem* elems, int n, int b);
    int insert(int pos, tElem elem);
    tElem erase(int pos);
    tElem get_value(int pos);
    int length();
    void clear();
}
```

Dónde `tListadeLista` corresponde a una lista que almacena listas. Para la implementación de la lista, usted debe implementar el TDA de Lista enlazada o Lista basada en arreglo visto en clases.

PISTA: deberá implementar dos tipos listas, uno que permita almacenar listas y otro que permita almacenar el tipo `tElem`.

Usted debe implementar las siguientes funciones **dentro** del *TDA*:

- `DeBlock(tElem* elems, int n, int b)`: constructor que inicializa la estructura con un arreglo de tamaño `n` utilizando bloques de tamaño `b`.
- `int insert(int pos, tElem elem)`: inserta `elem` en la posición `pos`. Retorna 1 si el elemento se insertó correctamente, en caso contrario se retorna 0.
- `tElem get_value(int pos)`: retorna el elemento almacenado en la posición `pos`.
- `int length()`: retorna el número de elementos en la estructura.

Puede agregar todas las funciones que usted estime conveniente.

3.1.2. Entrada

La entrada se realizará a través de un archivo llamado `pruebas.txt`. Este archivo tendrá el siguiente formato:

1. Dos enteros n y b, correspondientes a la cantidad de números iniciales en la secuencia y al tamaño de los bloques descritos anteriormente.
2. Luego, le sigue una línea con n enteros separados por un espacio, correspondientes a los números de la secuencia original.

3. Luego, le sigue un entero q, correspondiente a la cantidad de operaciones que se realizarán. Finalmente, están las q operaciones que podrán tener los siguientes formatos:
    - `I i v`: corresponde a insertar el elemento `v` en la posición `i`.
    - `G i`: corresponde a mostrar por pantalla el elemento en la posición `i`.
    - `L`: corresponde a que se debe mostrar por pantalla la cantidad de elementos en la estructura.

Un ejemplo del archivo de entrada es el siguiente:

```
7 3
1 3 2 5 6 2 3
8
G 0
I 0 4
G 0
L
G 1
G 2
I 4 4
L
```

3.1.3. Salida

La salida de datos se hará a través de la salida estándar (o sea por pantalla). Esta deberá mostrar lo que corresponda según las operaciones realizadas.

La salida correspondiente al ejemplo planteado anteriormente es la siguiente:

```
1
1
4
8
1
3
1
9
```

NOTA: su estructura no solo deberá poder almacenar enteros, sino que también debería ser capaz de almacenar cualquier tipo de dato.

4. Entrega de la Tarea

Entregue la tarea enviando un archivo comprimido `tarea1-apellido1-apellido2-apellido3.zip` o `tarea1-apellido1-apellido2-apellido3.tar.gz` (reemplazando sus apellidos según corresponda) a la página aula.usm del curso, a más tardar el día 31 de octubre, 2023, a las 23:59:00 hs (Chile Continental), el cual contenga como mínimo:

- Los archivos con los códigos fuentes necesarios para el funcionamiento de la tarea. ¡Los archivos deben compilar!
- **nombres.txt**, Nombre, ROL, Paralelo y qué programó cada integrante del grupo.
- **README.txt**, Instrucciones de compilación en caso de ser necesarias, y la forma de compilación que usó para cada problema de la tarea (debe ser alguna de las indicadas en los tutoriales entregados en Aula USM).

- El archivo `DeBlock.cpp` que contiene la implementación de la estructura.
- El archivo `test.cpp` que contiene la función main que permita ejecutar lo pedido.

5. Restricciones y Consideraciones

- Por cada día de atraso en la entrega de la tarea se descontarán 10 puntos en la nota.
- El plazo máximo de entrega es 5 días después de la fecha original de entrega.
- **Las tareas que no compilen no serán revisadas y serán calificadas con nota 0**.
- Debe usar **obligatoriamente** alguna de las formas de compilación indicadas en los tutoriales entregados en Aula USM.
- Por cada *Warning* en la compilación se descontarán 5 puntos.
- Si se detecta **COPIA** la nota automáticamente sera 0 (CERO), para todos los grupos involucrados. El incidente será reportado al jefe de carrera.
- La prolijidad, orden y legibilidad del código fuente es obligatoria. Habrá descuentos si alguno de estos items no se cumple.

6. Consejos de Programación

El código fuente del programa debe estar estructurado adecuadamente en archivos (separados de ser necesario). Si el código fuente está desordenado, se pueden descontar hasta 20 puntos de la nota.

Cada función programada debe tener comentarios de la siguiente forma:

```
/*****
*   TipoFunción NombreFunción
******
*   Resumen Función
******
*   Input:
*       tipoParámetro NombreParámetro : Descripción Parámetro
*        .......
******
*   Returns:
*       TipoRetorno, Descripción retorno
*****/
```

Por cada comentario faltante, se restarán 5 puntos.

Por último, la indentación (1 TAB o 4 espacios), es muy importante. Por **cada bloque mal indentado, se quitarán 10 puntos**.