Estructuras de datos - 2 parcial 2025s1
RONDA

Una ronda es una estructura de datos que permite modelar diferentes dominios. Un ejemplo es el de realizar juegos que involucran a varios jugadores, y donde cada uno juega en un turno determinado pasando al siguiente o al anterior según las reglas (por ej, el juego del Uno es uno de tales juegos). Se supone que los valores en la ronda son elementos abstractos que se pueden distinguir por un identificador, pero por simplicidad aquí los representaremos con un **int** (que representa solamente a dicho identificador). El tipo y la interfaz quedan definidos por al menos las siguientes declaraciones, que se ubican en un archivo **.h** (pueden existir otras operaciones que no son relevantes a esta evaluación)

```
struct RondaStr;
typedef RondaStr * Ronda;
```

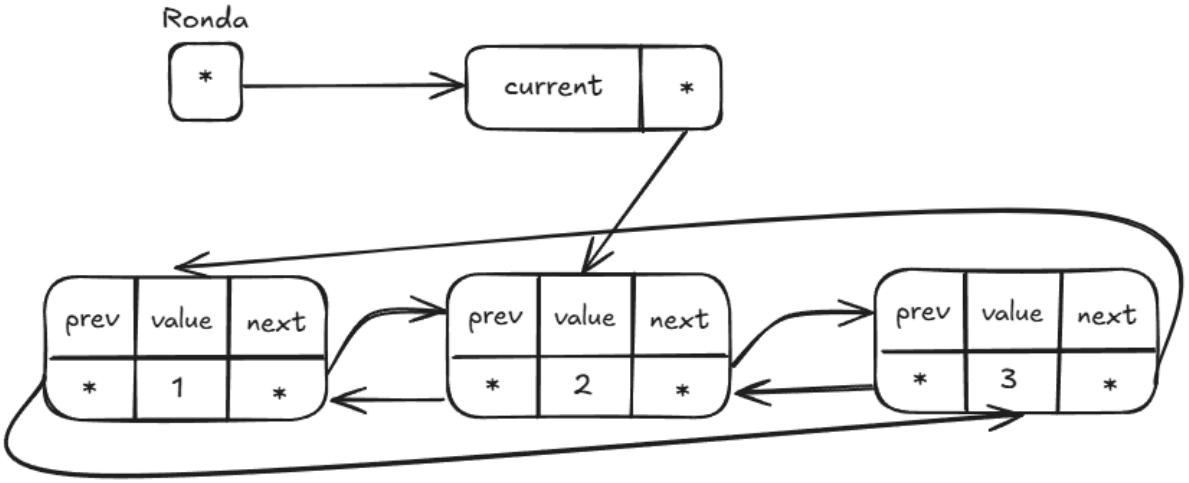
. Ronda mkRonda(), que crea una ronda sin elementos
. Int current(Ronda ronda), que retorna el elemento actual de la ronda, suponiendo que la ronda no está vacía. Falla si está vacía.
. void move(Int pos, Ronda ronda), que mueve el elemento actual la cantidad indicada de posiciones hacia adelante o hacia atrás.
. void insert(Int value, Ronda ronda), que inserta un elemento como el nuevo elemento actual de la ronda dada. Si la ronda ya tiene elementos, el anterior valor actual se mueve a la posición previa.
. void remove(Ronda ronda) que, suponiendo de que la lista no está vacía, remueve el elemento actual, moviendo el valor anterior a la posición actual. Si la lista está vacía, no hace nada
. Int length(Ronda ronda), que indica la cantidad de elementos en la ronda
AYUDA: utilizar el elemento actual para detectar cuándo terminar.

Para implementar el TAD se utilizaron los siguientes tipos de representación, que se incluyen en el archivo .**cpp**

```
struct RondaNode {
  Int value;
  RondaNode * next;
  RondaNode * prev;
}

struct RondaStr {
  RondaNode * current
}
```

La representación es similar a las listas encadenadas pero con doble encadenamiento. Cada nodo conoce a su anterior y a su siguiente. En el siguiente gráfico se muestra la estructura de una ronda con 3 elementos donde el actual es el número 2.



Se proveen implementadas algunas funciones para comprender la representación

```
Ronda mkRonda(){
  RondaStr * ronda = new RondaStr;
  ronda -> current = NULL;
  return ronda
}

Int current(Ronda ronda){
  return ronda -> current -> value;
}

void move(int pos, Ronda ronda){
  if(ronda -> current == NULL) { return }
  if(pos > o) {
    for(int i=0; i<pos; i++){
      ronda -> current = ronda -> current -> next;
  } else {
    for(int i=0; i>pos; i--){
      ronda -> current = ronda -> current -> prev;
    }
  }
}
```

EJERCICIOS:

Importante: En todos los ejercicios se evalúa el uso eficiente de recursos así como el análisis y justificación de costos.

1 - Escribir las invariantes de representación para poder mantener la coherencia del TAD en la representación

2 - Implementar las operaciones **insert**, **remove** y **length** del TAD ronda

3 - Implementar las siguientes funciones como usuario del TAD ronda:

a - **int * toArray(Ronda ronda)**, que construye una array con todos los valores de la ronda dada

b - **Ronda fromArray(int len, int * arra),** que construye una ronda con todos los valores del array dado. El elemento actual del resultado debe ser el primer elemento del array dado.