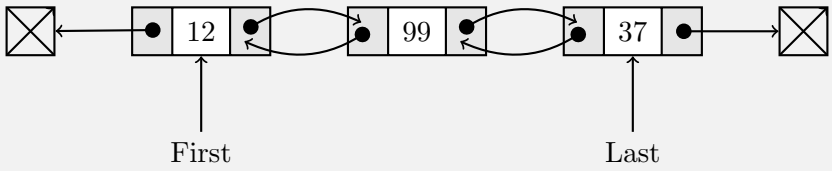
1 Primer parcial

1.1 DATE

Pregunta 1: Listas doblemente enlazadas. Una implementación alternativa de una secuencia es una lista doblemente enlazada. Este tipo de listas es diferente de las que vimos en clase porque que cada nodo tiene un puntero adicional que contiene la dirección del nodo anterior. En caso de ser una lista vacía los apuntadores al inicio y al final de la lista simplemente tienen el valor `NULL`. La siguiente figura presenta una lista doblemente enlazada que representa la secuencia $\langle 12, 99, 37 \rangle$.



Especifique las implementaciones de las siguientes operaciones de una lista doblemente enlazada indicando para cada una de ellas su complejidad.

1. (0.25 pts) Constructor para crear una lista vacía.
2. (0.25 pts) Constructor de copia y destructor.
3. (1 pts) Operación para remover de la lista el elemento en una posición dada:

```
void remove(int pos) {
  // ... su codigo aqui
}
```

4. (1 pts) Operación que remueve el último elemento.

Pregunta 2: Polinomios. Un polinomio es una combinación lineal de potencias enteras de una variable. Su forma general es: $P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x^1 + a_0 x^0$. La siguiente es una definición posiblemente *incompleta* de un producto (o termino) de

polinomio:

```
class Product {
private:
  int coefficient;
  int power;
public:
  Producto(int c, int p)
    : coefficient(c), power(p) {}
  int getCoefficient(void) const { return coefficient; }
  int getPower(void) const { return power; }
};
```

Por ejemplo, el término $5x^4$ de un polinomio sería representado por un objeto `Product t(5,4);`.
Con base en las definiciones anteriores escriba una definición para el tipo abstracto de dato polinomio. En su definición **solamente** especifique:

1. (0.5 pts) Constructor que recibe una secuencia de productos representada por un iterador y crea un polinomio con ellos. Asuma que los productos en la secuencia están dados en orden decreciente respecto al exponente.
2. (0.25 pts) Destructor y constructor de copia. Note que la classe `Product` no cuenta con un constructor de copia, si requiere de uno para este punto por favor escribalo.
3. (0.25 pts) Escriba un programa que declare el polinomio $5x^4 + 3x^2$

Pregunta 3: Adición de polinomios (1 pts). Dados dos polinomios P y Q representados por:

$$\begin{aligned} P(x) =& a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x^1 + a_0 x^0 \\ Q(x) =& b_n x^n + b_{n-1}x^{n-1} + \cdots + b_1 x^1 + b_0 x^0 \end{aligned}$$

Su suma es el polinomio

$$R(x) = (a_n + b_n)x^n + (a_{n-1} + b_{n-1})x^{n-1} + \cdots + (a_1 + b_1)x^1 + (a_0 + b_0)x^0$$

Escriba un método que calcule la suma de polinomios:

```
Polinomio add(const Polinomio& p) const {
  // Su implementacion aqui
}
```

Pregunta 4: Grado de un polinomio (0.5 pts). Para la clase polinomio que usted ha especificado en la pregunta 2, escriba un método que calcule el grado del polinomio.

1.2 DATE

Pregunta 5: Conjuntos. Después de haber viso las estructuras de datos básicas que representan secuencias, se abre un gran número de posibilidades para representar otras. Un conjunto en las matemáticas es una *colección* de elementos (ver el ejemplo 1). A continuación le propongo una implementación, *totalmente incompleta*, del tipo abstracto de dato conjunto:

```
template<typename E>
class Set {
private:
  Sequence<E> s;
public:
  Sequence(void) : s() {}
  bool member(const E& elem) const {
    // Su implementacion
  }
  void insert(const E& elem) {
    // Su implementacion
  }
  Set<E> Union(const Set<E>& x) const {
    // Su implementacion
  }
  Set<E> intersect(cons Set<E>& x) const {
    // Su implementacion
  }
};
```

En la implementación usted puede observar que el conjunto es representado por una secuencia que almacena sus elementos. Esta secuencia puede ser cualquiera de las vistas en clase: una lista o un vector. Por favor elija una de esas dos antes de responder las siguientes preguntas y tenga en cuenta su decisión al hacerlo[a].

1. Escriba la implementación del método `member`. Este método recibe un elemento y retorna si el elemento pertenece o no al conjunto. Asuma que el operador `==` está presente entre los elementos del tipo `E` y retorna si dos objetos de este tipo son iguales o no.

2. Implemente el método `insert` que recibe un elemento y lo inserta en el conjunto.

3. Implemente los métodos `Union` e `intersect` que implementan las operaciones de

unión e intersección de conjuntos. Note que estos métodos retornan los conjuntos resultantes de las operaciones.

4. Para cada uno de los puntos anteriores proporcione una cota de complejidad temporal y espacial (si es relevante).

Ejemplo 1: Recordatorio de operaciones en conjuntos. Tenga en cuenta las siguientes propiedades de los conjuntos que utilizan conjuntos de números enteros solo por razones únicamente didácticas.

- No existe la noción de orden en los elementos de un conjunto: $\{1, 2, 3, 4\} = \{4, 3, 2, 1\} = \{4, 1, 3, 2\}$
- Unión de conjuntos: $\{1, 2, 3, 4\} \cup \{4, 5, 6\} = \{1, 2, 3, 4, 5, 6\}$
- Intersección de conjuntos: $\{1, 2, 3, 4\} \cap \{4, 5, 6\} = \{4\}$

[a] No ser consistente al utilizar un solo tipo de dato tiene como penalización el 50% de los puntos obtenidos

1.3 Date

Pregunta 6: Diferencias entre listas y vectores (1.0 punto). Especifique la complejidad espacial y temporal de cada una de las siguientes operaciones tanto para vectores como para listas.

1. Borrar el último elemento
2. Retornar un elemento en una posición dada
3. Retornar el número de elementos en la estructura
4. Destruir la estructura de datos
5. Constructor de copia

Pregunta 7: Invertir una secuencia (2 puntos). Escriba el método *invertir* para cada una de las posibles representaciones de secuencias (lista y vector) vistas en clase. Para cada implementación proponga una cota de complejidad temporal. Tenga en cuenta que invertir **no** debe retornar una nueva secuencia; en su lugar, debe actuar sobre la secuencia representada por el objeto.

Pregunta 8: El programa misterio (2 puntos). Considere el siguiente método de la clase `List`.

```
List<T> misterio(void) const {
  Queue<T> q;
  Node *i = first_;
  while(i != NULL) {
    q.enqueue(i->getData());
    i = i->getNext();
  }
  Stack<T> s;
  while(!q.empty()) {
    s.push(q.front());
    q.dequeue();
  }
  List<T> l;
  while(!s.empty()){
    l.push_back(s.top());
    s.pop();
  }
  return l;
}
```

1. Describa con sus propias palabras qué hace el método. Tenga en cuenta que no se le pide relatar el algoritmo con sus palabras. Por ejemplo, usted puede decir: *el método suma todos los elementos de la lista.*

2. Proponga una cota de complejidad para el método.