Liste

1. Sovraccaricare l'operatore != in modo che l'espressione x != y restituisca true se e solo se le due liste x e y sono diverse
2. Definire e implementare nell'interfaccia i seguenti metodi:
    1. `clear()`: ende la lista vuota
    2. `num_elements(p1, p2)`: calcola il numero di elementi compresi fra le posizioni p1 e p2
    3. `exchange(p1, p2)`: scambia l'elemento in posizione p1 con quello in posizione p2
    4. `move_min_max()`: quando la lista è di elementi di tipo intero, sposta, nel modo più efficiente possibile, il massimo in ultima posizione, e il minimo in prima posizione
    5. Scrivere una funzione che elimini da una lista L1 tutti gli elementi presenti nella lista L2.
        - Se |L1|~|L2|=n, qual'è la complessità del vostro algoritmo?
3. Implementare il metodo Lista::insertionSort che ordina la lista utilizzando l'algoritmo di ordinamento per inserzione (Nota: non cancellare ne creare nuovi nodi)
4. Cambiare la realizzazione del metodo `Lista::canclista` in modo che dimezzi la dimensione del vettore quando il numero di elementi della lista è minore di dimensione/2
5. **[Polinomi]** Un polinomio di grado d può essere rappresentato con una lista lineare di d+1 elementi, in cui ogni elemento rappresenta un coefficiente del polinomio (l'elemento i-esimo della lista è il coefficiente dell'i-esimo termine, c_i x^d, del polinomio). Definire una classe C++ polynomial che metta a disposizione i seguenti metodi
    - polynomial() - crea un polinomio di grado zero
    - grado() - restituisce il grado di un polinomio
    - input(inStream) - acquisisce un polinomio dallo stream di input inStream. Possiamo assumere che l'input sia il grado del polinomio seguito dai valori dei coefficienti.
    - output(outStream) - visualizza il polinomio
    - somma(b) - somma con il polinomio b
    - moltiplica(b) - moltiplica con il polinomio b
    - valore(x) - restituisce il valore del polinomio nel punto x