Data Structures - L1:
Dynamic Array

Implement in C++ the given **container** (ADT) using a given representation and a **dynamic array** as a data structure. You are not allowed to use the *vector* from STL or from any other library.

Obs:

- You are not allowed to use the functions *memcpy* and *realloc*, because it is not safe to use *memcpy* and *realloc* on memory that was allocated with *new*. Also, if the memory location contains objects, undefined behavior can occur. The implementation might still work with these functions, but it is not a good practice to use them.
- If you think that you need helper functions, feel free to add them (but as *private* functions, so as to avoid violating the principle of abstraction) – this is valid for all labs during the semester, not just the current one.
- The solution for the first problem (0. **ADT Bag**) has been implemented as an example for you. You can find it in:
  Class Materials → Labs as the archive *BagRepresentedOnDynamicArray.zip*
- In *Class Materials → Labs* you can find a document named *Support material for labs - C++ basic syntax.pdf* that explains basic C++ syntax needed for the lab projects.

0. **ADT Bag** - represented using a dynamic array of elements. The array can contain duplicate elements.
1. **ADT Matrix** – represented as a sparse matrix, using a dynamic array of triples <line, column, value> (value ≠ 0), ordered lexicographically considering the <line, column> of every element.
2. **ADT Bag** – represented using a dynamic array of <element, frequency> pairs (or two dynamic arrays). For example, the bag [5, 10, -1, 2, 3, 10, 5, 5, -5] will be represented as [(5,3), (10, 2), (-1, 1),(2, 1), (3, 1),(-5, 1)].
3. **ADT SortedBag** – having elements of type **TComp**, represented using a dynamic array of <element, frequency> pairs (or two dynamic arrays), sorted using a relation on the elements.
4. **ADT SortedBag** – having elements of type **TComp**, sorted using a relation on the elements and stored in a dynamic array.
5. **ADT Set** – represented as a dynamic array of elements.
6. **ADT SortedSet** – having elements of type **TComp**, sorted using a relation on the elements and stored in a dynamic array.
7. **ADT Stack** – represented on a dynamic array.
8. **ADT Map** – represented as a dynamic array of <key, value> pairs.
9. **ADT Sorted Map** – represented as a dynamic array of <key, value> pairs, sorted using a relation over the keys.