Задача 1-1 (30 баллов). Покажите, как *деамортизировать* операции вставки в конец вектора, т.е. добиться того, чтобы операции добавления в конец и чтения элемента по индексу требовали $O(1)$ времени в *худшем* случае. Совет: по мере добавления новых элементов необходимо параллельно копировать уже имеющийся массив в массив увеличенного размера. Делать это следует с такой скоростью, чтобы в тот момент, когда меньший массив окажется заполнен, мы могли за время $O(1)$ выполнить переключение на новый массив.

Решение. Для деамортизации операций вставки в конец динамического массива и обеспечения того, чтобы добавление элемента и чтение по индексу имели сложность $O(1)$ в худшем случае, используем стратегию параллельного копирования.

1. Процесс начинается с аллокации 2х массивов с константными длинами k и $k * 2$, где $k >= 2$ и k - четное. Обозначим меньший массив за S, а больший за L.

2. В каждый момент времени имеем два массива с константной `capacity` n и $2n$ у S и L соответственно. Пользователь взаимодействует с S. Пока $\texttt{size}_S <$ половины $\texttt{capacity}_S$, делаем `push` лишь в S.

3. Как только $\texttt{size}_S$ достигает половины $\texttt{capacity}_S$, делаем такой же `push` в S. Создаем `index`$= 0$ на элементы массива S и делаем их `push` в массив L дважды, увеличивая `index`.

4. Когда S достигает своей $\texttt{capacity}_S$, "переключаем"пользователя на L, который к этому моменту имеет все элементы S и достиг половины $\texttt{capacity}_L$. Теперь $S := L$, старый S деаллоцируется и новый L с удвоенной $\texttt{capacity}_S$ аллоцируется. Здесь считаем, что операции аллокации и деаллокации памяти константны.

Благодаря этому процессу переключение между старым и новым массивом происходит плавно и не требует $O(n)$ времени в момент переключения. Таким образом, использование параллельного копирования позволяет деамортизировать вставку в конец вектора, обеспечивая гарантированное время $O(1)$ на каждую операцию добавления, даже в момент изменения емкости массива.