Задача 1: Реализирайте метод на едносвързания списък, който обръща реда на елементите му.

Пример:

```
ForwardList<int> myList = { 10, 20, 30, 40 };
mylist.reverse();

// After that myList contains: { 40, 30, 20, 10 }
// in that order.
```

Задача 2: Реализирайте метод на едносвързания списък, който премахва дублираните поредни елементи.

Пример:

```
ForwardList<int> myList =
                    { 10, 30, 20, 20, 40, 20, 30, 30, 10, 10, 10 };
myList.remove_repeats();

// After that myList contains: { 10, 30, 20, 40, 20, 30, 10 }
// in that order.
```

Задача 3: Реализирайте метод на едносвързания списък, който премахва повтарящите се елементи.

Пример:

```
ForwardList<int> myList =
                    { 10, 30, 20, 20, 40, 20, 30, 30, 10, 10, 10 };
myList.unique();

// After that myList contains: { 10, 30, 20, 40 }
// in that order.
```

Задача 4: Реализирайте метод append за едносвързания списък, който по подаден друг едносвързан списък го "залепва" за края на текущия.

Задача 5: Реализирайте метод `divide` за едносвързания списък, който по подаден предикат (булева функция) разделя елементите в списъка на две части. Елементите, които не удовлетворяват предиката да са в началото на списъка, а тези, които го удовлетворяват - в края.

Забележка:

Нека пренареждането да запазва релативната позиция на елементите в двете части.

Изисквания за сложност:

по памет: $O(1)$ *// in place*

по време: $O(n)$

Пример:

```
bool isEven( int num ) { return num % 2 == 0 };
// auto isEven = []( int num ) -> bool { return num % 2 == 0; }
ForwardList<int> mylist = { 5, 2, 7, 9, 1, 0, 3, 8, 4 };
mylist.divide( isEven );

// After that myList contains: { 5, 7, 9, 1, 3, 2, 0, 8, 4 }
// in that order.
```