Задача 2. Списки

В задачах ниже речь идет о связных списках, реализовать которые необходимо самостоятельно (можно пользоваться заготовками из примеров к лекциям). Если явно не указано иное, подразумеваются односвязные списки.

Список должен быть реализован в виде отдельного класса (отдельный класс для списка и отдельный – для элементов списка, последний может быть внутренним). В классе списка присутствуют методы создания списка (конструктор, методы добавления/удаления элементов и т.п.), перебора элементов списка, а также методы для решения непосредственно вашего варианта данной задачи.

Решение задачи предполагает наличие оконного интерфейса для создания первоначального списка (в том числе в виде чтения значений списка из текстового файла) и распечатку списка после выполненных преобразований (или ответа, если по условию задачи модифицировать список не требуется).

Если в условии задачи что-то непонятно – попросить пояснить преподавателя.

Варианты:

1. Для односвязного списка реализовать методы, осуществляющие циклический сдвиг элементов списка на k позиций влево или вправо.

2. Создать список, содержащий n первых чисел Фибоначчи.

3. Подсчитать кол-во максимальных элементов списка.

4. Для списка чисел найти количество локальных максимумов и локальных минимумов.

5. В двусвязном списке целых чисел перед и после каждого простого числа вставить новые элементы со значением 0.

6. Удалить из списка вещественных чисел элементы с порядковыми номерами с n по k включительно. Требуется модифицировать существующий список, а не создавать новый.

7. Удалить из списка чисел все элементы с четными значениями. Требуется модифицировать существующий список, а не создавать новый.

8. Удалить из односвязного списка чисел все элементы, содержащие минимальные или максимальные значения. Требуется модифицировать существующий список, а не создавать новый.

9. В списке (односвязном) целых чисел для каждого элемента, содержащего простое число, удалить элементы перед и после данного элемента, если они, в свою очередь, не являются простыми числами. Требуется модифицировать существующий список, а не создавать новый.

10. Дан список целых чисел. Создать новый список, содержащий неповторяющиеся значения из исходного списка. Первоначальный список не изменять.

11. Удалить из списка все элементы, индекс которых (отсчет начинаем с 0) соответствует какому-либо числу Фибоначчи. Требуется модифицировать существующий список, а не создавать новый.

12. Вставить после первого самого большого элемента в списке целых чисел элемент, содержащий то же самое значение (наибольшее), но с противоположным знаком. Задача должна быть решена в один проход по списку, в ходе которого необходимо найти и запомнить элемент (не значение, а именно элемент) с наибольшим значением.

13. Дан список целых чисел. Из каждых двух элементов данного списка оставить один, записав в него сумму двух элементов.

14. Дан список слов. Из каждой группы подряд идущих одинаковых слов оставить только одно. Обратите внимание, необходимо модифицировать существующий список, а не создавать новый.

15. Дан список из строк. Выбрать и вернуть в виде нового списка те строки, в которых содержится заданная подстрока. Исходный список поменяться не должен (т.е. для нового списка надо создать новые элементы списка).

16. Дан текстовый файл. Распечатать слова, имеющие максимальную длину (с повторениями, если они есть). Необходимо все слова из файла поместить в список. Далее пройти по этому списку, запоминая максимальную длину слова и помещая такие слова во второй список (если встречается новое более длинное слово, то второй список очищается и в него помещается новое самое длинное слово).

17. Дан список чисел. Найти и вернуть элемент (ListNode<T> / ListItem<T>) с положительным значением, после которого следует максимальное кол-во подряд идущих элементов с отрицательными значениями. В случае нескольких подходящих элементов вернуть последний. Реализовать в один проход по списку.

18. Удалить из списка строк все повторения, кроме первого. Дополнительных структур данных не использовать.

19. Дан список слов. Поменять местами первый и второй элемент списка, третий и четвертый и т.д. Обратите внимание: менять значение (Value) в элементах списка запрещено, необходимо именно переставить элементы (т.е. поменять ссылки Next и Prev у всех элементов списка).

20. Считалочка. N ребят расположены по кругу. (Каждому присвоен номер по порядку). Начав отсчёт от первого, удаляют каждого k-ого, смыкая при этом круг. Определить номер последнего, оставшегося в круге. ($k \leq N$).
    Указание: для решения задачи использовать список, в котором ссылочное поле последнего элемента содержит ссылку на первый элемент.

21. Дано два списка чисел, отсортированных по возрастанию. Необходимо объединить два эти списка в один итоговый список так, чтобы числа в списке также оказались отсортированными по возрастанию. Новых объектов ListNode / ListItem – не создавать (после выполнения операции объединения исходные списки окажутся пустыми.). По каждому из 2-х исходных списков можно пройти не более одного раза.

22. Для двусвязного списка написать процедуру проверки, что все ссылки на элементы списка расставлены правильно (если для А следующий элемент – В, то тогда для В предыдущий элемент – А). Продемонстрировать правильность проверки на позитивных (правильных) примерах и негативных (неправильных).

23. Для двухсвязного списка, для которого хранятся ссылки на первый и последний элемент и кол-во элементов, реализовать оптимальный метод поиска k-го по счету (начиная с начала списка) элемента. Надо проверить, k-ый элемент ближе к началу или к концу списка, и «добираться» до данного элемента, начиная или с начала или с конца.

24. Удалить из первого списка все элементы, которые встречают во втором списке. . Требуется модифицировать существующий список, а не создавать новый.

25. (*) В списке целых чисел поменять местами первый элемент, содержащий наименьшее значение, с последним элементом, содержащим наибольшее значение. Обратите внимание: менять значение (Value) в элементах списка запрещено, необходимо именно переставить элементы. Допускается не более одного прохода по списку.

26. Реализовать процедуру случайного перемешивания элементов списка.
    Подсказка: пройти по исходному списку и формировать новый список, где позиция очередного элемента определяется случайным образом (с помощью Random). Сложность алгоритма будет $O(n^2)$.

27. (*) В двусвязном списке поменять порядок элементов (1-ый становится последним, 2-ой – предпоследним и т.д.). Обратите внимание: менять значение (Value) в элементах списка запрещено, необходимо именно переставить элементы (т.е. поменять ссылки Next и Prev у всех элементов списка).

28. (*) Написать рекурсивную реализацию переворачивания (1-ый элемент становится последним, 2-ой – предпоследним и т.д.) односвязного списка. Новых объектов ListNode / ListItem не создавать.

29. В списке целых чисел переставить элементы списка таким образом, чтобы все отрицательные элементы оказались в начале списка, все положительные – в конце (нулевые – между положительными и отрицательными). При этом взаимный порядок следования отрицательных элементов, также как и положительных, поменяться не должен. Новых объектов ListNode / ListItem не создавать. Пройти по исходному списку можно только один раз.
    Подсказка: при проходе по списку необходимо отрицательные элементы объединять в один список, нулевые элементы – в другой, положительные – в третий, а затем полученные три списка объединить в один.

30. Дан список вещественных чисел $a_1, a_2, ..., a_n$. Сформировать новый список $b_1, b_2, ..., b_n$ размерности n по следующему правилу: элемент b_k равен сумме элементов исходного списка с номерами от 1 до k.

31. (*) Реализовать для списка сортировку слиянием (Merge Sort). Новых объектом ListNode / ListItem не создавать, манипулировать существующими объектами.

32. Дан список строк, состоящий из 3*n элементов. Переставить элементы исходного списка таким образом, чтобы новый порядок элементов был следующим (в квадратных скобках приведены номера первоначального списка, нумерация с 0): [0], [n], [2n], [1], [n+1], [2n+1], [2], [n+2], [2n+2] и т.д. Новых объектов ListNode / ListItem не создавать.
    Подсказка: первоначально найти элементы [0], [n], [2n], затем синхронно продвигаясь по элементам исходного списка с этих позиций формировать новый список.
    Пример: { 1, 2, 3, 44, 55, 66, 777, 888, 999 } –> { 1, 44, 777, 2, 55, 888, 3, 66, 999 }

33. (*) Рекурсивная реализация проверки, является ли односвязный список палиндромом. Подсказка: одним из вариантов может быть функция/метод которая проверяет проверяет равенство элементов на «обратном» ходе рекурсии. При вызове этой функции в качестве параметра передается объект left (например, массив из одного элемента), первоначально содержащий ссылку на первоначальный элемент списка. После того, как функция дойдет до последнего элемента списка, происходит сравнение последнего элемента списка с элементом в left, после чего происходит «переход» к следующему (второму) элементу списка в left и возврат из функции. Таким образом, когда мы будем сравнивать предпоследний элемент, left будет указывать на второй элемент и т.д.

34. (*) Найти самую длинную последовательность в первом списке Xs, состоящую из элементов второго списка. Результат должен быть возвращен в виде итератора / итерируемого объекта, к которому можно было бы применить цикл for each, например так:

```
for (Integer v : list1.longestSubSeqOfAnotherList(list2)) { … }
```

35. (*) Реализовать для списка пузырьковую сортировку:
https://ru.wikipedia.org/wiki/Сортировка_пузырьком).
Новых объектов ListNode / ListItem не создавать.
Обмениваться должны сами элементы, а не их значения (требование, конечно, нелогичное, исключительно для того, чтобы задачу сделать более сложной / интересной).

36. (*) Реализовать для списка сортировку вставками:
https://ru.wikipedia.org/wiki/Сортировка_вставками)
Новых объектов ListNode / ListItem не создавать.

37. (*) Дан список студентов (свойства: фио, курс). Модифицировать список таким образом, чтобы вначале шли студенты 1-го курса, затем 2-го, далее 3-го и т.д. (упорядочивать студентов по фамилии в рамках одного курса не требуется). Новых объектов ListNode / ListItem – не создавать.
Будем считать, что точно известно, что курс может быть от 1 до 6.
Подсказка: для каждого курса завести 2 переменных – для первого и последнего элемента (можно использовать массив). Далее пройти по переданному списку, переприсоединяя очередной элемент к нужному списку (соответствующему курсу студента). И в конце объединить полученные списки по курсам в один (при этом не забыть учесть, что на каком-то курсе может вообще не оказаться студентов).

38. Реализовать удаление из списка значений, которые встречаются в списке только один раз. Дополнительных структур данных не создвать, работать только со списком. Для каждого элемента списка придется пройти по списку отдельно, таким образом сложность будет $O(n^2)$.

39. Реализовать удаление из списка значний, которые встречаются в списке более одного раза. Дополнительных структур данных не создвать, работать только со списком. Для каждого элемента списка придется пройти по списку отдельно, таким образом сложность будет $O(n^2)$.

40.