Для решения поставленных задач был использован диалект языка Лисп – newLISP.

**Задача №1 (Lab3-1.lsp):**

Идея метода заключается в сравнение разделенных на группы элементов последовательности, находящихся друг от друга на некотором расстоянии. Количество шагов сортировки и шаги в данной программе вычисляются методом, предложенным Дональдом Кнутом.

Случаи, когда список пустой и список содержит 1 элемент рассматриваются отдельно, так как эти списки уже отсортированы и не требуют никаких изменений, поэтому они просто выводятся.

Тестовые наборы данных и результаты приведены в Таблице 1:

Таблица 1. Тестовые набора данных

|  |  |
| --- | --- |
| Тестовые данные | Результат |
| ‘ () | () |
| ‘(5) | (5) |
| ‘ (10 -2) | (-2 10) |
| ‘(-2 1 -6 0) | (-6 -2 0 1) |
| ‘ (3 6 2 9 1 6 0 4 -2 40 -3 1 7 0 -13 34 74 23 84 -34 12 4 8 34 64 22 -66 32) | (-66 -34 -13 -3 -2 0 0 1 1 2 3 4 4 6 6 7 8 9 12 22 23 32 34 34 40 64 74 84) |

**Задача №2 (Lab3-2-1.lsp):**

Сортировка методом прямого выбора заключается в том, что в неотсортированном списке ищется локальный минимум, далее найденный минимум меняется местами с первым элементом в списке, после чего эти действия повторяются пока не пройдут все элементы списка.

В программе за поиск локального минимума отвечает функция find-min. Чтобы удалить элемент списка из исходного списка используется функция remove-item. Итоговый список формируется в функции select-sort.

Тестовые наборы данных и результаты приведены в Таблице 2:

Таблица 2. Тестовые набора данных

|  |  |
| --- | --- |
| Тестовые данные | Результат |
| ‘ () | () |
| ‘(5) | (5) |
| ‘ (10 -2) | (-2 10) |
| ‘(-2 1 -6 0) | (-6 -2 0 1) |
| ‘ (3 6 2 9 1 6 0 4 -2 40 -3 1 7 0 -13 34 74 23 84 -34 12 4 8 34 64 22 -66 32) | (-66 -34 -13 -3 -2 0 0 1 1 2 3 4 4 6 6 7 8 9 12 22 23 32 34 34 40 64 74 84) |

**Задача №3 (Lab3-3.lsp):**

В данных программе проверяются головы списков: меньшее значение головы списка передается в результирующий список, и вызывается рекурсивно исходная функция, при этом список, в котором оказалось, наименьшее значение головы передается не полностью, а только его хвост. Рекурсия заканчивается, когда один из списков становится пустым.

Тестовые наборы данных и результаты приведены в Таблице 3:

Таблица 3. Тестовые набора данных

|  |  |
| --- | --- |
| Тестовые наборы данных | Результат |
| '(-3 -4 0 2 4 5 7) '(-14 -9 -2 0 2 4 6 10) | (-14 -9 -3 -4 -2 0 0 2 2 4 4 5 6 7 10) |
| '() '(-14 -9 -2 0 2 4 6 10) | (-14 -9 -2 0 2 4 6 10) |
| '(-3 -4 0 2 4 5 7) '() | (-3 -4 0 2 4 5 7) |

**Задача №4 (Lab3-4-1.lsp):**

В данной программе удаляются все подсписки на указанном уровне, если подсписков на указанном уровне нет, то выводится исходный список. Сначала функция находит нужный нам уровень (до тех пор, пока не выполнится условие (= level 1)). После этого если на этом уровне встретится подсписок, то мы его пропускаем, если атомарный элемент, то мы его добавляем в результирующий список.

Тестовые наборы данных и результаты приведены в Таблице 4:

Таблица 4. Тестовые набора данных

|  |  |
| --- | --- |
| Тестовые наборы данных | Результат |
| '(1 2 3 (1 2 3 (4 5 6) 7 8 (2 (2 4) 3 4) 9) 5) 1 | (1 2 3 5) |
| '(1 2 3 (1 2 3 (4 5 6) 7 8 (2 (2 4) 3 4) 9) 5) 2 | (1 2 3 (1 2 3 7 8 9) 5) |
| '(1 2 3 (1 2 3 (4 5 6) 7 8 (2 (2 4) 3 4) 9) 5) 3 | (1 2 3 (1 2 3 (4 5 6) 7 8 (2 3 4) 9) 5) |
| '(1 2 3 (1 2 3 (4 5 6) 7 8 (2 (2 4) 3 4) 9) 5) 4 | (1 2 3 (1 2 3 (4 5 6) 7 8 (2 (2 4) 3 4) 9) 5) |

**Задача №5 (Lab3-2-2.lsp):**

Отличительной особенностью сортировки Хоара является операция разбиения массива на две части относительно опорного элемента. Например, если последовательность требуется упорядочить по возрастанию, то в левую часть будут помещены все элементы, значения которых меньше значения опорного элемента, а в правую элементы, чьи значения больше или равны опорному. Вне зависимости от того, какой элемент выбран в качестве опорного, массив будет отсортирован, но все же наиболее удачным считается ситуация, когда по обеим сторонам от опорного элемента оказывается примерно равное количество элементов. Если длина какой-то из получившихся в результате разбиения частей превышает один элемент, то для нее нужно рекурсивно выполнить упорядочивание, то есть повторно запустить алгоритм на каждом из отрезков.

В программе в качестве опорного элемента выбирается голова списка. В функции list< происходит рекурсивная сортировка элементов списка, которые больше опорного элемента. В функции list>= происходит рекурсивная сортировка элементов списка, которые меньше или равны опорному элементу.

Тестовые наборы данных и результаты приведены в Таблице 5:

Таблица 5. Тестовые набора данных

|  |  |
| --- | --- |
| Тестовые данные | Результат |
| ‘ () | () |
| ‘(1) | (1) |
| ‘ (2 1) | (1 2) |
| ‘ (2 1 3) | (1 2 3) |
| ‘ (1 2 6 23 -1 1 3 1 3 4 5 6 8 9 12 3 5 6 89 1 2 4 5 7 2 3 0 1) | (-1 0 1 1 1 1 1 2 2 2 3 3 3 3 4 4 5 5 5 6 6 6 7 8 9 12 23 89) |

**Задача №6 (Lab3-4-2.lsp):**

В данных программе, если список не содержит подсписков, то результатом будет 0. Если же список содержит подсписки, это проверяется наличием не атомарных элементов с помощью функции atom?, то в найденном подсписке проверяется наличие подсписков и так далее. В итоге выводится наибольшая глубина списка.

Тестовые наборы данных и результаты приведены в Таблице 6:

Таблица 6. Тестовые набора данных

|  |  |
| --- | --- |
| Тестовые наборы данных | Результат |
| '(1 2 3) | 0 |
| '(1 (2) 3) | 1 |
| '(1 (3 (2)) 3) | 2 |
| '(1 (2 (2 (3))) ((3 (2 (3))))) | 4 |