Общая постановка задачи

- Опишите функцию, выполняющую обработку, описанную в задании с номером вашего варианта.
- Приведите набор тестовых вызовов описанной функции, демонстрирующих все варианты ее работы.
- Опишите программу в текстовом файле с именем `task11-NN.lsp`, где `NN` — номер вашего варианта. Полученный файл загрузите на портал в качестве выполненного задания.

Пример выполнения задания

ЗАДАННОЕ ВЫРАЖЕНИЕ: *Фермеру в двуместной лодке нужно переправиться на другой берег с козлом, капустой и волком, причем в отсутствие фермера козел неравнодушен к капусте, а волк с удовольствием съест козла.*

РЕШЕНИЕ: Содержимое файла `task11-NN.lsp`:

```
1  ;; составление состояния
2  ;; состояние составляется в форме (farmer wolf goat cabbage)
3  ;; где farmer - берег, на котором находится феремер (e или w)
4  ;; где wolf - берег, на котором находится волк (e или w)
5  ;; где goat - берег, на котором находится коза (e или w)
6  ;; где cabbage - берег, на котором находится капуста (e или w)
7  (defun make-state (farmer wolf goat cabbage)
8    (list farmer wolf goat cabbage))
9
10 ;; набор функций, для определения берега на котором находится
11 ;; соответствующее действующее лицо в состоянии state
12 ;; (результат - e или w)
13 (defun farmer-side (state) (nth 0 state))
14 (defun wolf-side (state) (nth 1 state))
15 (defun goat-side (state) (nth 2 state))
16 (defun cabbage-side (state) (nth 3 state))
17
```

```
18 ;; проверка на то, что состояние state является безопасным
19 ;; (никто никого и ничего не съест)
20 ;; результат - state, если state безопасно
21 ;; nil - если state небезопасно
22 (defun safe (state)
23   (if (not (eq (goat-side state) (farmer-side state)))
24     (cond
25       ((eq (wolf-side state) (goat-side state)) nil)
26       ((eq (cabbage-side state) (goat-side state)) nil)
27       (T state))
28     state))
29
30 ;; переход из состояния state в состояние, когда фермер один
31 ;; переправляется на другой берег
32 (defun farmer-takes-self (state)
33   (safe (make-state
34             (opposite (farmer-side state))
35             (wolf-side state)
36             (goat-side state)
37             (cabbage-side state))))
38
39 ;; переход из состояния state в состояние, когда фермер с волком
40 ;; переправляется на другой берег
41 (defun farmer-takes-wolf (state)
42   (cond
```

```
43      ((equal (farmer-side state) (wolf-side state))
44         (safe (make-state (opposite (farmer-side state))
45                           (opposite (wolf-side state))
46                           (goat-side state)
47                           (cabbage-side state))))
48      (t nil)))
49
50  ;; переход из состояния state в состояние, когда фермер с козой
51  ;; переправляется на другой берег
52  (defun farmer-takes-goat (state)
53    (cond
54      ((equal (farmer-side state) (goat-side state))
55         (safe (make-state (opposite (farmer-side state))
56                           (wolf-side state)
57                           (opposite (goat-side state))
58                           (cabbage-side state))))
59      (t nil)))
60
61  ;; переход из состояния state в состояние, когда фермер с капустой
62  ;; переправляется на другой берег
63  (defun farmer-takes-cabbage (state)
64    (cond
65      ((equal (farmer-side state) (cabbage-side state))
66         (safe (make-state (opposite (farmer-side state))
67                           (wolf-side state)
```

```
68                              (goat-side state)
69                              (opposite (cabbage-side state)))))
70      (t nil)))
71
72 ;; функция выдает противоположный берег для берега side
73 ;; выдает (e или w)
74 (defun opposite (side)
75   (cond
76     ((equal side 'e) 'w)
77     ((equal side 'w) 'e)))
78
79 ;; основная функция - поиск пути от начального состояния state
80 ;; к конечному состоянию goal со списком уже пройденных состояний been-list
81 (defun path (state goal been-list)
82   (cond
83     ((null state) nil)
84     ((equal state goal) (reverse (cons state been-list)))
85     ((not (member state been-list :test #'equal))
86        (or
87          (path (farmer-takes-self state)
88                goal (
89                cons state been-list))
90          (path (farmer-takes-wolf state)
91                goal (cons state been-list))
92          (path (farmer-takes-goat state)
```

```
93                  goal
94                  (cons state been-list))
95            (path (farmer-takes-cabbage state)
96                  goal
97                  (cons state been-list))))))
98
99  ;; пример запуска
100 (print (path '(w w w w) '(e e e e) ()))
```

Варианты заданий

1. *К реке подъехали 4 рыцаря с оруженосцами и обнаружили одну трехместную лодку. Как им переправиться на другой берег, если все оруженосцы наотрез отказались оставаться в обществе незнакомых рыцарей?*

2. *Три миссионера и три каннибала должны пересечь реку в лодке, в которой могут поместиться только двое. Миссионеры должны соблюдать осторожность, чтобы каннибалы не получили на каком-либо берегу численное преимущество. Как переплыть реку?*

3. *Нужно переправить через реку с помощью одного плота семью (мать, отца, 2-х дочерей и 2-х сыновей) и полицейского с заключенным.*

Следует соблюдать следующие правила:

1. На плоту могут одновременно перемещаться максимум 2 человека.

2. Папе не разрешается находиться с дочерьми без присутствия матери.

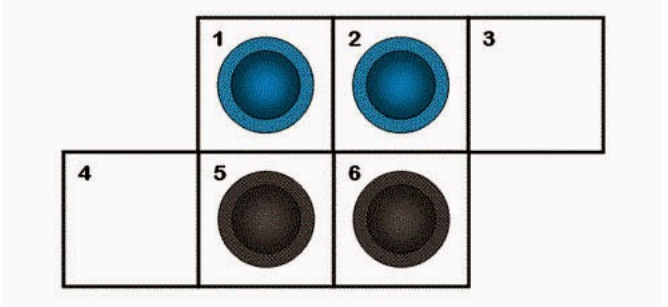
3. Маме не разрешается находиться с сыновьями без присутствия отца.

4. Заключённого нельзя оставлять без полицейского ни с одним из членов семьи.

5. Управлять плотом могут только полицейский и родители.

4. *Человеку в трехместной лодке нужно переправиться на другой берег с козлом, капустой, двумя волками и собакой, причем собака в ссоре с волком, козел неравнодушен к капусте, а волк и собака не могут оставаться наедине с козлом.*

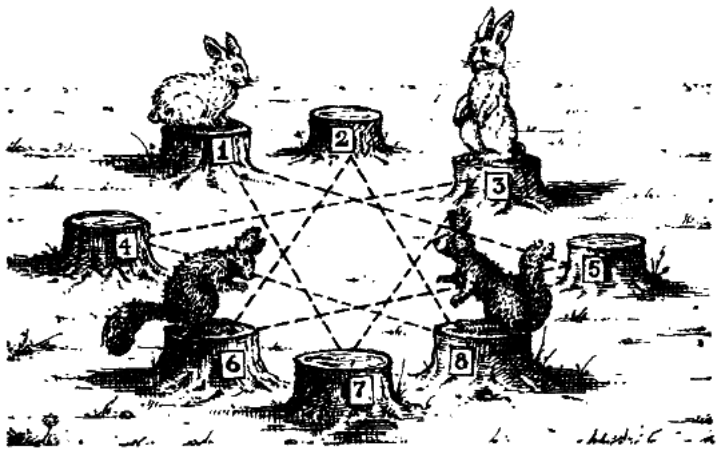
5. *Нужно поменять местами синие и черные фишки. Разрешается двигать фишки только на смежное пустое место.*



6. *Есть восемь перенумерованных пней. На пнях 1 и 3 сидят кролики, на пнях 6 и 8 — белки. И белки, и кролики хотят обменяться пнями: белки желают сидеть на местах кроликов, а кролики — на местах белок. Попасть на новое место они могут, прыгая с пня на пень по следующим правилам:*

1. *прыгать с пня на пень можно только по тем линиям, которые показаны на рисунке; каждый зверек может делать несколько прыжков кряду;*

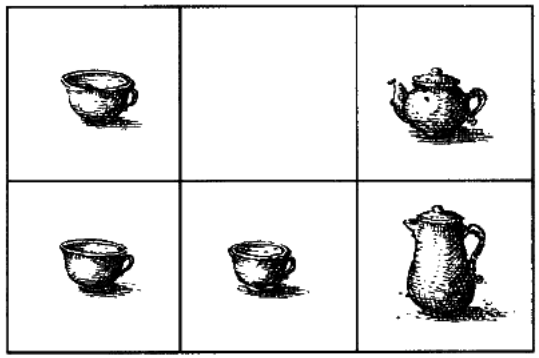
2. *два зверька на одном пне поместиться не могут, поэтому прыгать можно только на свободный пень.*



Нужно найти минимальную последовательность прыжков, для достижения цели.

7. *Стол разграфлен на 6 квадратов, в каждом из которых, кроме одного, помещается какой-нибудь предмет. Можно передвигать предметы из одного квадрата в другой по определенным правилам, а именно:*

1. *можно перемещать предмет только в тот квадрат, который окажется свободным;*
2. *нельзя передвигать предметы по диагонали квадрата;*
3. *нельзя переносить один предмет поверх другого;*
4. *нельзя также помещать в квадрат более одного предмета, даже временно.*



Нужно поменять местами чайник и молочник.

8. *Реализовать алгоритм решения задачи о поиске последовательности перемещений коня на шахматной доске из заданной начальной клетки в конечную.*

9. *Реализовать алгоритм решения задачи о поиске последовательности перемещений коня на шахматной доске размера $m \times n$ (например, 4×4 или 4×5) из*

заданной начальной клетки (нижняя левая клетка) в нее же, при этом надо побывать хотя бы по одному разу на всех остальных клетках доски.

10. *Магараджа — это шахматная фигура, сочетающая возможности ферзя и коня. Таким образом, магараджа может ходить и бить на любое количество клеток по диагонали, горизонтали и вертикали (т.е. как ферзь), а также либо на две клетки по горизонтали и на одну по вертикали, либо на одну по горизонтали и на две по вертикали (как конь).*

Найти число способов расставить на доске $N \times N$ ровно K магараджей так, чтобы они не били друг друга.

11. *Реализовать алгоритм решения задачи о ханойских башнях. Даны три стержня, на один из которых нанизаны восемь дисков, причем диски отличаются размером и лежат меньшее на большем. Задача состоит в том, чтобы перенести пирамиду из n дисков за наименьшее число ходов. За один раз разрешается переносить только один диск, причём нельзя класть больший диск на меньший.*

12. *Дан набор сосудов заданной вместимости и наполненности. Необходимо определить последовательность переливаний приводящую к необходимому количеству жидкости в каждом сосуде. За одно переливание можно наполнить до краев один сосуд из другого или полностью вылить всю жидкость из одного сосуда в другой.*

13. *Реализовать алгоритм решения задачи о сборке кубика Рубика размера* $2 \times 2 \times 2$.

14. *Реализовать алгоритм решения задачи о сборке кубика Рубика размера* $3 \times 3 \times 3$.

15. *Варикон — цилиндрическая башня с шестью вращающимися вокруг оси цилиндра «этажами». На каждом из первых пяти этажей — 4 окошка. В окошках — цветные шарики. Всего в вариконе по пять шариков четырех цветов. На шестом этаже — только одно окошко без шарика. Если повернуть этажи так, чтобы под/над окошком с шариком было пустое окошко, то в него можно переместить шарик.*
Для заданной начальной конфигурации варикона найти последовательность действий, приводящую варикон в состояние, когда каждый вертикальный ряд окошек содержит шарики только одного цвета.

16. *Реализовать алгоритм решения задачи коммивояжера, для минимизации пройденного им расстояния. Задана матрица расстояний между городами. Если между городами нет прямой дороги, соответствующее значение равно* -1.

17. *Транспортные маршруты представлены номерами со списками остановок, заданными в порядке их следования по маршруту. Реализовать алгоритм решения задачи о нахождении маршрутов для пассажира с заданной начальной остановки в заданную конечную.*

18. *В робототехнике одной из классических проблем является задача определения того, как робот должен перемещаться в окружающем пространстве, чтобы перейти от текущего положения в некоторую конечную позицию, при этом, например, избегая столкновений и/или минимизируя время движения. Реализовать алгоритм решения задачи о планировании грузового робота. Робот двигается в плоскости, движение осуществляется дискретно с шагом длины* 1*, по четырем направлениям север-юг-восток-запад. Робот может поднять одну ед. груза и опустить одну ед. груза (в любой соседней с ним клетке). На плоскости в некоторых ячейках (клетках) размера* 1×1 *сосредоточены грузы. Задача состоит в нахождении последовательности действий робота для перемещения всех грузов в заданную клетку. Начальное положение робота задано.*

19. *Граф задан парами* $(a\ b)$ *(из* a *можно попасть в* b*). Найти последовательность перемещений из начальной вершины в заданную.*

20. *Реализовать алгоритм решения задачи о перемещении цветных клеток.*

21. *В робототехнике одной из классических проблем является задача определения того, как робот должен перемещаться в окружающем пространстве, чтобы перейти от текущего положения в некоторую конечную позицию, при этом, например, избегая столкновений и / или минимизируя время движения. Рассмотрим планирование пути для «внедорожника»-робота-автомобиля в очень большом пространстве (карта части России или карта большого куска Марса),*

описанном в виде квадратной сетки. Мы предполагаем, что робот мал по сравнению с размером ячеек сетки, то есть он находится полностью в пределах квадрата сетки, и требуется некоторое время, чтобы преодолеть этот квадрат сетки. Каждой клетке сетки присвоено значение между 0 *и* 1*, означающее, сколько времени требуется, чтобы пройти через нее, или* -1*, означающее, что эта клетка сетки непроходима. Из каждой клетки робот может перемещаться только в соседние по горизонтали или вертикали клетки. Найти оптимальный маршрут из заданной позиции робота в клетку с заданными координатами.*

22. *Реализовать алгоритм C4.5 построения дерева решений. Вход — таблица. Выход — дерево.*

23. *Робот может передвигаться по плоскости, три направления движения (вперед, влево, вправо), шаг фиксирован (соседняя клетка). Может «брать» груз, стоящий перед ним, а также «класть» груз на клетку перед ним. Робот находится в комнате размером* $m \times m$ *клеток, задано его начальное положение, заданы клетки, в которые он не может встать, задана клетка, на которой находится груз, клетка, на которую надо перенести груз. Написать код для нахождения последовательности действий робота для переноса груза в нужную клетку.* $m = 10$*.*

24. *Задача планирования в мире блоков. Количество блоков* N *задано. Блоки — квадраты, каждый блок имеет номер (от 1 до* N*). Проведена прямая на плос-*

кости. Начальное состояние — блоки находятся в некотором порядке на плоскости. Блоки-квадраты могут «лежать» на прямой или «стоять» один над другим. Целевое состояние — блоки расположены вертикально один над другим по возрастанию их номеров. Можно снимать блок сверху (если он стоит на другом блоке) и ставить его на прямую слева или справа, двигать по прямой влево или вправо.

25. *Робот-пылесос: Постановка: есть некоторое пространство клеток, есть как пустые клетки, так и занятые. Роботу необходимо объехать все свободные клетки.*