УАСД - Втори Срок Контролно 1
Вариант 1, нечетни

Задача 1

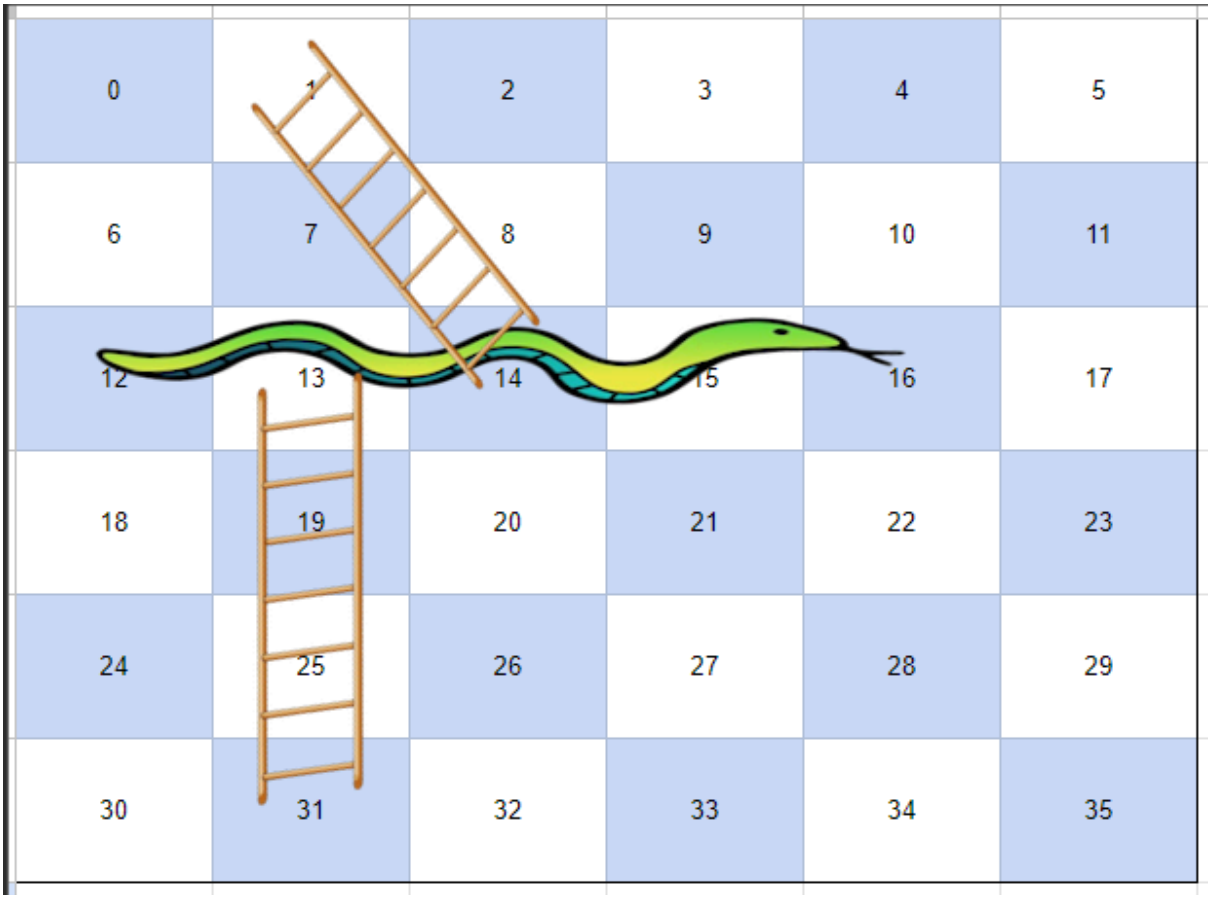
Дадена е NxN матрица номерирана от 0 до n^2 -1 започвайки отгоре в ляво. Дъската представлява игра на “Стълби и змии”. Ако на някое поле има стълба тя ви пренася напред, ако е змия ви връща назад. Започвате от поле 0. Всеки ход имате избор измежду 6 стъпки, симулирайки мятане на зар от 1 до 6 и всяка стъпка ви мести напред със 1-6 полета. Полетата, които нямат змия или стълба са отбелязани в матрицата с -1, а тези, които имат съдържат число от 0 до n^2 -1. След придвижване напред ако на достигнатото поле има змия или стълба тя задължително ви мести на съответното поле отбелязано в матрицата.

Да се изведе минималният брой движения нужни за да се достигне финалното поле n^2 -1. Ако това не е възможно да се изведе -1.

На поле 0 и n^2 -1 е гарантирано, че няма змия или стълба. Всеки ход може да минете през максимум една змия или стълба. Тоест ако след минаване по стълба, полето, което е достигнато има друга стълба или змия **не** продължавате по нея.

Може да използвате на готово имплементацията на двусвързан списък в dlist.c като опашка (insertBegin и pop).

Пример:



На тази картинка отговорът е 4.
- първи ход: 1 поле напред, което ни води 14 по стълбата
- втори ход: 2 полета напред до 16, което ни връща на 12 по змията
- трети ход: 1 поле напред до 13, което ни води по стълбата до 31
- четвърти ход: 4 полета напред до финала

Масива на това поле би изглеждало така:

```
[
        [-1, 14, -1, -1, -1, -1],
        [-1,  -1, -1, -1, -1, -1],
        [-1,  31, -1, -1, 12, -1],
        [-1,  -1, -1, -1, -1, -1],
        [-1,  -1, -1, -1, -1, -1],
        [-1,  -1, -1, -1, -1, -1],
]
```

Обяснете в коментар (дори и да сте успели да направите кода) каква е идеята на решението ви, как работи и каква е неговата сложност по време.

Задача 2

Нека си припомним как работеха алгоритмите на Дийкстра и Прим по двата псевдокода:

```
Prim(G, s):

for v in G:
    dist[v] = inf
    p[v] = Null

dist[s] = 0
S = {}

// докато не сме обходили всички върхове V.
while S != V:
    //търсим минималния измежду необходените
    x = find_min_vertex(G, S, dist)
    S.add(x)
    for y in G->adjList[x]:
        // w((x,y)) теглото на ребро (x,y)
        if dist[y] > w((x, y)):
            dist[y] = w((x, y))
            p[y] = x
```

```
Djikstra(G, s):

for v in G:
    dist[v] = inf
    p[v] = Null

dist[s] = 0
S = {}

// докато не сме обходили всички върхове V.
while S != V:
    //търсим минималния измежду необходените
    x = find_min_vertex(G, S, dist)
    S.add(x)
    for y in G->adjList[x]:
        // w((x,y)) теглото на ребро (x,y)
        if dist[y] > dist[x] + w((x, y)):
            dist[y] = dist[x] + w((x, y))
            p[y] = x
```

Обяснете стъпка по стъпка как ще протекат те върху графа на картинката започвайки от връх 1. Покажете какво е покриващото дърво, което се получава в двата случая и как то се репрезентира в масива p. Най-добре ще е на всяка стъпка да показвате как се променя този масив както и другия помощен масив dist, който при алгоритъма на Дийкстра съдържа информация за текущите разстояния от стартовия връх, а при този на Прим пази теглата на

текущите най-леки ребра свързващи върховете.