سوال ۱.

تابع $f(m,k,n)$ تعداد حالاتی را به دست می‌آورد که گاو با k بار عوض کردن جنس علف و محدودیت m کیلو برای هر نوع علف، n کیلو علف را می‌خورد.
$f(m,0,0)$ برابر است با ۱ (در اینصورت گاو با k بار تعویض علف n کیلو علف خورده است. اگر n منفی باشد یا k صفر باشد $f(m,n,k)$ صفر می‌شود

در غیر اینصورت داریم:

$$f(m,n,k) = \sum_{i=1}^{m} f(m,k-1,n-i)$$

Sudocode:

```
A = [][] // 2d Array with default value -1.
f(m, n, k):
    if n=0 and k=0 : return 1
    if n<0 or k=0 : return 0
    if A[k][n] ≠ -1 : return A[k][n]
    sum // value to store sum of loop (default value 0)
    for i in 1 to m do:
        sum = sum + f(m, k-1, n-i)
    A[k][n] = sum.
```

پیچیدگی این تابع $= O(m \times (k) \times n) = O(mkn)$

حلقه‌ی کد

کم کردن k تا جایی که صفر شود

کم کردن n تا جایی صفر یا منفی شود.

پیاده‌سازی بهینه‌تر در شکل زیر موجود است.

Optimized Sudo code

```
f(m, k, n):
A = [ ][ ] // 2D Array with default value 1.
A[0][0] = 1
for i in 1 to k do:
    for j in i to n do:
        A[i][j] = A[i][j-1] + A[i-1][j-1]
        if j - m - 1 >= 0 do:
            A[i][j] -= A[i-1][j-m-1]
return A[k][n]
```

پیچیدگی = $O(kn)$

سوال ۲.

فرض می کنیم تابع $L(i,j)$ طول بزرگترین زیررشته‌ی پالیندروم از اندیس i تا j را می‌دهد. تابع L را به صورت زیر تعریف می کنیم:

(کاراکتر i-ام رشته ←)

$$L(i,j) = \begin{cases} L(i+1, j-1) + ۲ & \text{if } a[i] = a[j] \\ \max\{L(i+1, j), L(i, j-1)\} & \text{otherwise} \end{cases}$$

حال یک ماتریس را در نظر می گیریم که شماره‌ی سطر آن اندیس شروع (کاراکتر اول) و شماره‌ی ستون آن اندیس پایان (کاراکتر آخر) رشته را نشان می دهد.

ماتریس را به کمک تابع بازگشتی و برنامه ریزی پویا محاسبه می کنیم. $L(0,n-1)$ درایه‌ی جواب مسئله‌ی ماست.

sudo code:

```
A = [][] // 2D Array used as matrix with default value 1.
for i in 0 to n do:
    for j in 0 to n-i+1 do:
        k = j + i - 1
        if a[j] = a[k] and j = 2. do:
            A[j][k] = 2
        elseif a[j] = a[k]
            A[j][k] = A[j+1][k-1] + 2
        else
            A[j][k] = max{A[j][k-1], A[j+1][k]}
```

پیچیدگی زمانی $= O((n+1) + (n) + (n-1) + \dots + 1) = O(n^۲)$

سوال ۳.

دو آرایه برای تاکسی ها در نظر می گیریم و در صورتی که هر تاکسی به درخواست خ ام پاسخ دهد خ وارد آرایه آن تاکسی می شود. (آرایه های A و B)

فرض می کنیم می خواهیم به درخواست خ-ام پاسخ دهیم. فرض می کنیم اعضای A دون p_{a_1} ، p_{a_2} ، ... ، p_{a_x} و اعضای B دون p_{b_1} ، p_{b_2} ، ... ، p_{b_y} قرار دارند. بدون اینکه به کلیت مسئله آسیبی وارد شود فرض می کنیم حالت بهینه حالتی است که p_i وارد A می شود و پس p_j قرار می گیرد. در نتیجه تمام p های بین خ و ز در B قرار می گیرند. پس آرایه ها به شکل زیر هستند.

A: $p_x \cdots p_j p_i$

B: $p_y \cdots p_k p_{j+1} p_{j+2} \cdots p_{i-1}$

$j+1 > k$ است. چون ز عنصری است که حالت بهینه به ازای آن محاسبه شده است و در بزرگترین اندیس است.

فرض کنید d_{ij} برابر مسافت طی شده جهت متداول طی کردن نقاط i تا j است. به عبارتی دیگر:

$$d_{ij} = x_{p_i} + x_{p_{i+1}} + \cdots + x_{p_{j-1}} + x_{p_j}$$

آرایه دو بعدی d را در $O(n^2)$ محاسبه می کنیم.

همچنین فرض می کنیم D_{ij} برابر فاصله بین نقطه i ام و j ام است و آن را برای هر دو i و j در $O(n^2)$ محاسبه می کنیم.

اگر کوتاه ترین گذر به ازای هر n را در آرایه M ذخیره کنیم $M[i]$ به شکل متناظر محاسبه می شود:

$$M[i] = M[j] + D_{k,j+1} + d_{j+1,i-1} + D_{j,i}$$

به این شکل $M[n]$ در $O(n^2)$ محاسبه می شود.

سوال ۴.

فرض می کنیم کمترین هزینه رفتن از نقطه ۱ به نقطه i برابر با A_i باشد. در این صورت، آخرین قطاری که ما را به مقصد (نقطه i) رسانده است یا از نقطه ۱ رسانده است یا ۲ یا ۳ یا ... یا $i-1$. پس برای هر i داریم:

$$A_i = \min_{j=1}^{i-1} \{A_j + a_{ji}\} \Rightarrow$$

$$A_1 = 0$$

$$A_2 = \min\{A_1 + a_{12}\} = a_{12}$$

$$A_3 = \min\{A_1 + a_{13}, A_2 + a_{23}\}$$

$$\vdots$$

$$A_n = \min\{A_1 + a_{1n} + A_2 + a_{2n} + \cdots + A_{n-1} + a_{n-1\,n}\}$$

Sudo code:

```
A = [-1 for i in 1 to max]

findMinOf(n): // function to find min
              // cost for going from 1 to n

    if A[n] is not equal to -1:
        return A[n]

    else:
        if n is equal to 1:
            return 0
        if n is equal to 2:
            return a12
        temp = []
        for i in 1 to n-1:
            append A[i] + a_in
        return minimum element of temp.
```

زمان اجرای الگوریتم برای رشته n = $O(1+2+\cdots+(n-1)) = O(n^2)$

سوال ۶.

طول بلندترین زیر رشته صعودی را پیدا می کنیم (یا نزولی)

Sudocode

```
f(Arr[], n)
  Arr = [ ]
  Arr[0] = 1
  for i in 1 to n-1
     Arr[i] = 1
     for j in 0 to i
        if Arr[i] > Arr[j] and Arr[i] < Arr[j]+1 : do
           Arr[i] = Arr[j] + 1
  return max{Arr[0], Arr+n}
```