A. Glory Addicts,

2 seconds, 512 megabytes, standard input, standard output

The hero is addicted to glory, and is fighting against a monster.

The hero has n skills. The i-th skill is of type a_i (either **fire** or **frost**) and has initial damage b_i.

The hero can perform all of the n skills in any order (with each skill performed exactly **once**). When performing each skill, the hero can play a magic as follows:

- If the current skill immediately follows another skill of a different type, then its damage is **doubled**.

In other words,

1. If a skill of type fire and with initial damage c is performed immediately after a skill of type fire, then it will deal c damage;
2. If a skill of type fire and with initial damage c is performed immediately after a skill of type frost, then it will deal $2c$ damage;
3. If a skill of type frost and with initial damage c is performed immediately after a skill of type fire, then it will deal $2c$ damage;
4. If a skill of type frost and with initial damage c is performed immediately after a skill of type frost , then it will deal c damage.

Your task is to find the **maximum** damage the hero can deal.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 10^5$) — the number of test cases. The following lines contain the description of each test case.

The first line of each test case contains an integer n ($1 \le n \le 10^5$), indicating the number of skills.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 1$), where a_i indicates the type of the i-th skill. Specifically, the i-th skill is of type fire if $a_i = 0$, and of type frost if $a_i = 1$.

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$), where b_i indicates the initial damage of the i-th skill.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output the maximum damage the hero can deal.

input
4 4 0 1 1 1 1 10 100 1000 6 0 0 0 1 1 1 3 4 5 6 7 8 3 1 1 1 1000000000 1000000000 1000000000 1 1 1

output
2112 63 3000000000 1

In the first test case, we can order the skills by $[3, 1, 4, 2]$, and the total damage is $100 + 2 \times 1 + 2 \times 1000 + 10 = 2112$.

In the second test case, we can order the skills by $[1, 4, 2, 5, 3, 6]$, and the total damage is $3 + 2 \times 6 + 2 \times 4 + 2 \times 7 + 2 \times 5 + 2 \times 8 = 63$.

In the third test case, we can order the skills by $[1, 2, 3]$, and the total damage is $1000000000 + 1000000000 + 1000000000 = 3000000000$.

In the fourth test case, there is only one skill with initial damage 1, so the total damage is 1.

B. Prefix Sum Addicts,

2 seconds, 512 megabytes, standard input, standard output

Suppose $a_1, a_2, \ldots, a_n$ is a sorted **integer** sequence of length n such that $a_1 \le a_2 \le \cdots \le a_n$.

For every $1 \le i \le n$, the prefix sum s_i of the first i terms $a_1, a_2, \ldots, a_i$ is defined by

$$s_i = \sum_{k=1}^{i} a_k = a_1 + a_2 + \cdots + a_i.$$

Now you are given the last k terms of the prefix sums, which are $s_{n-k+1}, \ldots, s_{n-1}, s_n$. Your task is to determine whether this is possible.

Formally, given k integers $s_{n-k+1}, \ldots, s_{n-1}, s_n$, the task is to check whether there is a sequence $a_1, a_2, \ldots, a_n$ such that

1. $a_1 \le a_2 \le \cdots \le a_n$, and
2. $s_i = a_1 + a_2 + \cdots + a_i$ for all $n - k + 1 \le i \le n$.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 10^5$) — the number of test cases. The following lines contain the description of each test case.

The first line of each test case contains two integers n ($1 \le n \le 10^5$) and k ($1 \le k \le n$), indicating the length of the sequence a and the number of terms of prefix sums, respectively.

The second line of each test case contains k integers $s_{n-k+1}, \ldots, s_{n-1}, s_n$ ($-10^9 \le s_i \le 10^9$ for every $n - k + 1 \le i \le n$).

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output "YES" (without quotes) if it is possible and "NO" (without quotes) otherwise.

You can output "YES" and "NO" in any case (for example, strings "yEs", "yes" and "Yes" will be recognized as a positive response).

input
4 5 5 1 2 3 4 5 7 4 -6 -5 -3 0 3 3 2 3 4 3 2 3 4

output
Yes Yes No No

In the first test case, we have the only sequence $a = [1, 1, 1, 1, 1]$.

In the second test case, we can choose, for example, $a = [-3, -2, -1, 0, 1, 2, 3]$.

In the third test case, the prefix sums define the only sequence $a = [2, 1, 1]$, but it is not sorted.

In the fourth test case, it can be shown that there is no sequence with the given prefix sums.

C. Even Number Addicts, 2 seconds, 512 megabytes, standard input, standard output

Alice and Bob are playing a game on a sequence $a_1, a_2, \ldots, a_n$ of length n. They move in turns and **Alice moves first**.

In the turn of each player, he or she should select an integer and remove it from the sequence. The game ends when there is no integer left in the sequence.

Alice wins if the sum of her selected integers is **even**; otherwise, Bob wins.

Your task is to determine who will win the game, if both players play optimally.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \leq t \leq 100$) — the number of test cases. The following lines contain the description of each test case.

The first line of each test case contains an integer n ($1 \leq n \leq 100$), indicating the length of the sequence.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$), indicating the elements of the sequence.

Output

For each test case, output "Alice" (without quotes) if Alice wins and "Bob" (without quotes) otherwise.

input
4 3 1 3 5 4 1 3 5 7 4 1 2 3 4 4 10 20 30 40

output
Alice Alice Bob Alice

In the first and second test cases, Alice always selects two odd numbers, so the sum of her selected numbers is always even. Therefore, Alice always wins.

In the third test case, Bob has a winning strategy that he always selects a number with the same parity as Alice selects in her last turn. Therefore, Bob always wins.

In the fourth test case, Alice always selects two even numbers, so the sum of her selected numbers is always even. Therefore, Alice always wins.

D. Permutation Addicts, 2 seconds, 512 megabytes, standard input, standard output

Given a permutation $a_1, a_2, \ldots, a_n$ of integers from 1 to n, and a threshold k with $0 \leq k \leq n$, you compute a sequence $b_1, b_2, \ldots, b_n$ as follows.

For every $1 \leq i \leq n$ in increasing order, let $x = a_i$.

- If $x \leq k$, set b_x to the last element a_j ($1 \leq j < i$) that $a_j > k$. If no such element a_j exists, set $b_x = n + 1$.
- If $x > k$, set b_x to the last element a_j ($1 \leq j < i$) that $a_j \leq k$. If no such element a_j exists, set $b_x = 0$.

Unfortunately, after the sequence $b_1, b_2, \ldots, b_n$ has been completely computed, the permutation $a_1, a_2, \ldots, a_n$ and the threshold k are discarded.

Now you only have the sequence $b_1, b_2, \ldots, b_n$. Your task is to find any possible permutation $a_1, a_2, \ldots, a_n$ and threshold k that produce the sequence $b_1, b_2, \ldots, b_n$. **It is guaranteed that** there exists at least one pair of permutation $a_1, a_2, \ldots, a_n$ and threshold k that produce the sequence $b_1, b_2, \ldots, b_n$.

A permutation of integers from 1 to n is a sequence of length n which contains all integers from 1 to n exactly once.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \leq t \leq 10^5$) — the number of test cases. The following lines contain the description of each test case.

The first line of each test case contains an integer n ($1 \leq n \leq 10^5$), indicating the length of the permutation a.

The second line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($0 \leq b_i \leq n + 1$), indicating the elements of the sequence b.

It is guaranteed that there exists at least one pair of permutation $a_1, a_2, \ldots, a_n$ and threshold k that produce the sequence $b_1, b_2, \ldots, b_n$.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output the threshold k ($0 \leq k \leq n$) in the first line, and then output the permutation $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$) in the second line such that the permutation $a_1, a_2, \ldots, a_n$ and threshold k produce the sequence $b_1, b_2, \ldots, b_n$. If there are multiple solutions, you can output any of them.

input
3 4 5 3 1 2 6 7 7 7 3 3 3 6 4 4 4 0 0 0

output
2 1 3 2 4 3 1 2 3 4 5 6 3 6 5 4 3 2 1

For the first test case, permutation $a = [1, 3, 2, 4]$ and threshold $k = 2$ will produce sequence b as follows.

- When $i = 1$, $x = a_i = 1 \leq k$, there is no a_j ($1 \leq j < i$) that $a_j > k$. Therefore, $b_1 = n + 1 = 5$.
- When $i = 2$, $x = a_i = 3 > k$, the last element a_j that $a_j \leq k$ is a_1. Therefore, $b_3 = a_1 = 1$.
- When $i = 3$, $x = a_i = 2 \leq k$, the last element a_j that $a_j > k$ is a_2. Therefore, $b_2 = a_2 = 3$.
- When $i = 4$, $x = a_i = 4 > k$, the last element a_j that $a_j \leq k$ is a_3. Therefore, $b_4 = a_3 = 2$.

Finally, we obtain sequence $b = [5, 3, 1, 2]$.
For the second test case, permutation $a = [1, 2, 3, 4, 5, 6]$ and threshold $k = 3$ will produce sequence b as follows.

- When $i = 1, 2, 3$, $a_i \leq k$, there is no a_j ($1 \leq j < i$) that $a_j > k$. Therefore, $b_1 = b_2 = b_3 = n + 1 = 7$.
- When $i = 4, 5, 6$, $a_i > k$, the last element a_j that $a_j \leq k$ is a_3. Therefore, $b_4 = b_5 = b_6 = a_3 = 3$.

Finally, we obtain sequence $b = [7, 7, 7, 3, 3, 3]$.
For the third test case, permutation $a = [6, 5, 4, 3, 2, 1]$ and threshold $k = 3$ will produce sequence b as follows.

- When $i = 1, 2, 3$, $a_i > k$, there is no a_j ($1 \leq j < i$) that $a_j \leq k$. Therefore, $b_4 = b_5 = b_6 = 0$.
- When $i = 4, 5, 6$, $a_i \leq k$, the last element a
  Therefore, $b_1 = b_2 = b_3 = a_3 = 4$.

Finally, we obtain sequence $b = [4, 4, 4, 0, 0, 0]$.

E. Balance Addicts,

2 seconds, 512 megabytes, standard input, standard output

Given an integer sequence $a_1, a_2, \ldots, a_n$ of length n, your task is to compute the number, modulo 998244353, of ways to partition it into several **non-empty continuous** subsequences such that the sums of elements in the subsequences form a **balanced** sequence.

A sequence $s_1, s_2, \ldots, s_k$ of length k is said to be *balanced*, if $s_i = s_{k-i+1}$ for every $1 \le i \le k$. For example, $[1, 2, 3, 2, 1]$ and $[1, 3, 3, 1]$ are balanced, but $[1, 5, 15]$ is not.

Formally, every partition can be described by a sequence of indexes $i_1, i_2, \ldots, i_k$ of length k with $1 = i_1 < i_2 < \cdots < i_k \le n$ such that

1. k is the number of non-empty continuous subsequences in the partition;
2. For every $1 \le j \le k$, the j-th continuous subsequence starts with a_{i_j}, and ends exactly before $a_{i_{j+1}}$, where $i_{k+1} = n + 1$. That is, the j-th subsequence is $a_{i_j}, a_{i_j+1}, \ldots, a_{i_{j+1}-1}$.

There are 2^{n-1} different partitions in total.
Let $s_1, s_2, \ldots, s_k$ denote the sums of elements in the subsequences with respect to the partition $i_1, i_2, \ldots, i_k$. Formally, for every $1 \le j \le k$,

$$s_j = \sum_{i=i_j}^{i_{j+1}-1} a_i = a_{i_j} + a_{i_j+1} + \cdots + a_{i_{j+1}-1}.$$

For example, the partition $[1 \mid 2, 3 \mid 4, 5, 6]$ of sequence $[1, 2, 3, 4, 5, 6]$ is described by the sequence $[1, 2, 4]$ of indexes, and the sums of elements in the subsequences with respect to the partition is $[1, 5, 15]$.

Two partitions $i_1, i_2, \ldots, i_k$ and $i'_1, i'_2, \ldots, i'_{k'}$ (described by sequences of indexes) are considered to be different, if at least one of the following holds.

- $k \ne k'$,
- $i_j \ne i'_j$ for some $1 \le j \le \min\{k, k'\}$.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 10^5$) — the number of test cases. The following lines contain the description of each test case.

The first line of each test case contains an integer n ($1 \le n \le 10^5$), indicating the length of the sequence a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$), indicating the elements of the sequence a.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, output the number of partitions with respect to which the sum of elements in each subsequence is balanced, modulo 998244353.

input

```
6
1
1000000000
2
1 1
4
0 0 1 0
5
1 2 3 2 1
5
1 3 5 7 9
32
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

output

```
1
2
3
4
2
150994942
```

For the first test case, there is only one way to partition a sequence of length 1, which is itself and is, of course, balanced.

For the second test case, there are 2 ways to partition it:

- The sequence $[1, 1]$ itself, then $s = [2]$ is balanced;
- Partition into two subsequences $[1 \mid 1]$, then $s = [1, 1]$ is balanced.

For the third test case, there are 3 ways to partition it:

- The sequence $[0, 0, 1, 0]$ itself, then $s = [1]$ is balanced;
- $[0 \mid 0, 1 \mid 0]$, then $s = [0, 1, 0]$ is balanced;
- $[0, 0 \mid 1 \mid 0]$, then $s = [0, 1, 0]$ is balanced.

For the fourth test case, there are 4 ways to partition it:

- The sequence $[1, 2, 3, 2, 1]$ itself, then $s = [9]$ is balanced;
- $[1, 2 \mid 3 \mid 2, 1]$, then $s = [3, 3, 3]$ is balanced;
- $[1 \mid 2, 3, 2 \mid 1]$, then $s = [1, 7, 1]$ is balanced;
- $[1 \mid 2 \mid 3 \mid 2 \mid 1]$, then $s = [1, 2, 3, 2, 1]$ is balanced.

For the fifth test case, there are 2 ways to partition it:

- The sequence $[1, 3, 5, 7, 9]$ itself, then $s = [25]$ is balanced;
- $[1, 3, 5 \mid 7 \mid 9]$, then $s = [9, 7, 9]$ is balanced.

For the sixth test case, every possible partition should be counted. So the answer is $2^{32-1} \equiv 150994942 \pmod{998244353}$.

F. Connectivity Addicts,

2 seconds, 512 megabytes, standard input, standard output

This is an interactive problem.

Given a simple undirected graph with n vertices numbered from 1 to n, your task is to color all the vertices such that for every color c, the following conditions hold:

1. The set of vertices with color c is **connected**;
2. $s_c \le n_c^2$, where n_c is the number of vertices with color c, and s_c is the sum of degrees of vertices with color c.

It can be shown that there always exists a way to color all the vertices such that the above conditions hold.
Initially, you are only given the number n of vertices and the degree of each vertex.

In each query, you can choose a vertex u. As a response, you will be given the k-th edge incident to u, if this is the k-th query on vertex u.

You are allowed to make **at most n queries**.

An undirected graph is simple if it does not contain multiple edges or self-loops.

The degree of a vertex is the number of edges incident to it.

A set S of vertices is connected if for every two different vertices $u, v \in S$, there is a path, which only passes through vertices in S, that connects u and v. That is, there is a sequence of edges $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$ with $k \ge 1$ such that

1. $u_1 = u$, $v_k = v$, and $v_i = u_{i+1}$ for every $1 \le i < k$; and
2. $u_k \in S$ and $v_k \in S$ for every $1 \le i \le k$.

Especially, a set containing only one vertex is connected.

Interaction

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases. The following lines contain the description and the interactive section of each test case.

For each test case, you begin the interaction by reading an integer n ($1 \le n \le 1000$) in the first line, indicating the nu
graph.

The second line contains n integers $d_1, d_2, \ldots, d_n$ ($0 \le d_i \le n-1$), where d_i is the degree of vertex i.

To make a query on vertex u ($1 \le u \le n$), you should output

- "? u"

in a separate line. If this is the k-th query on vertex u, vertex $e_{u,k}$ will be given in the next separate line, where $(u, e_{u,k})$ is the k-th edge incident to vertex u. In case of $k > d_u$, define $e_{u,k} = -1$. You should make **no more than** n "?" queries.
To give the answer, you should output

- "! c_1 c_2 ... c_n"

in a separate line, where c_i ($1 \le c_i \le n$) is the color of vertex i. After that, your program should continue to the next test case, or terminate if this is the last test case.
It is guaranteed that the graph is a simple undirected graph.

It is guaranteed that the sum of n over all test cases does not exceed 1000.

In case your query format is invalid, or you have made more than n "?" queries, you will receive **Wrong Answer** verdict.

After printing a query, do not forget to output end of line and flush the output. Otherwise, you will get Idleness limit exceeded. To do this, use:

- fflush(stdout) or cout.flush() in C++;
- System.out.flush() in Java;
- flush(output) in Pascal;
- stdout.flush() in Python;
- see documentation for other languages.

Hack Format

The first line of the hack contains an integer t ($1 \le t \le 1000$), indicating the number of test cases. The following lines contain the description of each test case.

The first line of each test case contains an integer n ($1 \le n \le 1000$), indicating the number of vertices in the graph.

Then n lines follow. The i-th line contains an integer d_i ($0 \le d_i \le n-1$), indicating the degree of vertex i, and then d_i **distinct** integers $e_{i,1}, e_{i,2}, \ldots, e_{i,d_i}$ ($1 \le e_{i,j} \le n$ and $e_{i,j} \ne i$), where $(i, e_{i,j})$ is the j-th edge incident to vertex i.

It should be guaranteed that the graph is a simple undirected graph.

It should be guaranteed that the sum of n over all test cases does not exceed 1000.

input

```
1
5
2 2 2 2 0

2

4

2

4
```

output

```
? 1

? 1

? 3

? 3

! 1 1 2 2 3
```

In the example, there is only one test case.

In the test case, there are $n = 5$ vertices with vertices $1, 2, 3, 4$ of degree 2 and vertex 5 of degree 0. It is obvious that vertex 5 is isolated, i.e., it does not connect to any other vertices.

A possible interaction is shown in the sample input and output, where 4 "?" queries are made on vertex 1 twice and vertex 3 twice. According to the responses to these queries, we know that each of vertex 1 and vertex 3 connects to two vertices 2 and 4.

A possible solution is shown in the sample output, where vertex 1 and vertex 2 are colored by 1, vertex 3 and vertex 4 are colored by 2, and vertex 5 is colored by 3. It can be seen that this solution satisfies the required conditions as follows.

- For color $c = 1$, vertex 1 and vertex 2 are connected. Moreover, $n_1 = 2$ and $s_1 = d_1 + d_2 = 2 + 2 = 4 \le n_1^2 = 2^2 = 4$;
- For color $c = 2$, vertex 3 and vertex 4 are connected. Moreover, $n_2 = 2$ and $s_2 = d_3 + d_4 = 2 + 2 = 4 \le n_2^2 = 2^2 = 4$;
- For color $c = 3$, there is only one vertex (vertex 5) colored by 3. Moreover, $n_3 = 1$ and $s_3 = d_5 = 0 \le n_3^2 = 1^2 = 1$.

G. Anti-Increasing Addicts,

2 seconds, 512 megabytes, standard input, standard output

You are given an $n \times n$ grid.

We write (i, j) to denote the cell in the i-th row and j-th column. For each cell, you are told whether yon can delete it or not.

Given an integer k, you are asked to delete **exactly** $(n-k+1)^2$ cells from the grid such that the following condition holds.

- You cannot find k not deleted cells $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$ that are strictly increasing, i.e., $x_i < x_{i+1}$ and $y_i < y_{i+1}$ for all $1 \le i < k$.

Your task is to find a solution, or report that it is impossible.

Input

Each test contains multiple test cases. The first line contains an integer t ($1 \le t \le 10^5$) — the number of test cases. The following lines contain the description of each test case.

The first line of each test case contains two integers n and k ($2 \le k \le n \le 1000$).

Then n lines follow. The i-th line contains a binary string s_i of length n. The j-th character of s_i is 1 if you can delete cell (i, j), and 0 otherwise.

It's guaranteed that the sum of n^2 over all test cases does not exceed 10^6.

Output

For each test case, if there is no way to delete exactly $(n-k+1)^2$ cells to meet the condition, output "NO" (without quotes).

Otherwise, output "YES" (without quotes). Then, output n lines. The i-th line should contain a binary string t_i of length n. The j-th character of t_i is 0 if cell (i, j) is deleted, and 1 otherwise.

If there are multiple solutions, you can output any of them.

You can output "YES" and "NO" in any case (for example, strings "yEs", "yes" and "Yes" will be recognized as a positive response).

input

```
4
2 2
10
01
4 3
1110
0101
1010
0111
5 5
01111
10111
11011
11101
11110
5 2
10000
01111
01111
01111
01111
```

output

```
YES
01
11
YES
0011
1111
1111
1100
NO
YES
01111
11000
10000
10000
10000
```

For the first test case, you only have to delete cell $(1, 1)$.

For the second test case, you could choose to delete cells $(1, 1)$, $(1, 2)$, $(4, 3)$ and $(4, 4)$.

For the third test case, it is no solution because the cells in the diagonal will always form a strictly increasing sequence of length 5.

H. Palindrome Addicts, 2 seconds, 1024 megabytes, standard input, standard output

Your task is to maintain a queue consisting of lowercase English letters as follows:

- "push c": insert a letter c at the back of the queue;
- "pop": delete a letter from the front of the queue.

Initially, the queue is empty.

After each operation, you are asked to count the number of **distinct** palindromic substrings in the string that are obtained by concatenating the letters from the front to the back of the queue.

Especially, the number of distinct palindromic substrings of the empty string is 0.

A string $s[1..n]$ of length n is palindromic if $s[i] = s[n - i + 1]$ for every $1 \leq i \leq n$.

The string $s[l..r]$ is a substring of string $s[1..n]$ for every $1 \leq l \leq r \leq n$.

Two strings $s[1..n]$ and $t[1..m]$ are distinct, if at least one of the following holds.

- $n \neq m$;
- $s[i] \neq t[i]$ for some $1 \leq i \leq \min\{n, m\}$.

Input

The first line is an integer q ($1 \leq q \leq 10^6$), indicating the number of operations.

Then q lines follow. Each of the following lines contains one of the operations as follows.

- "push c": insert a letter c at the back of the queue, where c is a lowercase English letter;
- "pop": delete a letter from the front of the queue.

It is guaranteed that no "pop" operation will be performed when the queue is empty.

Output

After each operation, print the number of distinct palindromic substrings in the string presented in the queue.

input

```
12
push a
pop
push a
push a
push b
push b
push a
push a
pop
pop
pop
push b
```

output

```
1
0
1
2
3
4
5
6
5
4
3
4
```

Let s_k be the string presented in the queue after the k-th operation, and let c_k be the number of distinct palindromic substrings of s_k. The following table shows the details of the example.

k	s_k	c_k
1	a	1
2	empty	0
3	a	1
4	aa	2
5	aab	3
6	$aabb$	4
7	$aabba$	5
8	$aabbaa$	6
9	$abbaa$	5
10	$bbaa$	4
11	baa	3
12	$baab$	4

It is worth pointing out that

- After the 2-nd operation, the string is empty and thus has no substrings. So the answer is 0;
- After the 8-th operation, the string is "$aabbaa$". The 6 distinct palindromic substrings are "a", "aa", "$aabbaa$", "$abba$", "b", and "bb".