【模拟】-敏感字段加密

题目描述与示例

题目描述

给定一个由多个命令字组成的命令字符串；

1. 字符串长度小于等于 `127` 字节，只包含**大小写字母，数字，下划线和偶数个双引号**
2. 命令字之间以一个或多个下划线 `_` 进行分割
3. 可以通过两个双引号 `""` 来标识包含下划线 `_` 的命令字或空命令字（仅包含两个双引号的命令字），双引号不会在命令字内部出现

请对指定索引的敏感字段进行加密，替换为 `******`（6 个 `*`），并删除命令字前后多余的下划线 `_`。如果无法找到指定索引的命令字，输出字符串 `ERROR`

输入描述

输入为两行 第一行为命令字索引 `K`（从 `0` 开始） 第二行为命令字符串 `S`

输出描述

输出处理后的命令字符串 如果无法找到指定索引的命令字，输出字符串 `ERROR`

示例一

输入

```
1 1
2 password_a12345678_timeout_100
```

输出

```
password_******_timeout_100
```

说明

示例二

输入

```
2
aaa_password_"a12_45678"_timeout_100_""_
```

输出

```
aaa_password_******_timeout_100_""
```

说明

`"a12_45678"` 为包含双引号的命令字，需要整体替换。原字符串中末尾的 `"_"` 需要删除。

解题思路

纯模拟题。

简单理解题意就是，把下划线 `_` 作为分割符对原字符串进行分割（如果产生空字符串则跳过），将分割后索引为 `K` （索引从 `0` 记起）的字符串修改为 `******` 。

题意中比较难理解的一句话是

> - 可以通过两个双引号 `""` 来标识包含下划线 `_` 的命令字或空命令字（仅包含两个双引号的命令字），双引号不会在命令字内部出现

这句话是意思是，如果某一个下划线 `_` 出现在一对引号内，那么**这个下划线则不作为分割符使用**。

这一点也导致我们不能够直接使用 `split()` 来处理原字符串，而是需要在一个 `for` 循环中来进行字符串的分割。

其基本过程如下。

1. 由于需要考虑某段字符串是否位于一对引号内，我们可以设置一个标志 `flag` 。当
   - `flag` 为 `True` 时，说明此时位于一对引号内
   - `flag` 为 `False` 时，说明此时不位于一对引号内

   初始化这个 `flag` 为 `False` ，并在循环过程中反复切换 `flag` 。
2. 构建一个初始化列表 `lst = [""]` ，包含一个空串。 `lst` 用于储存最终的分割结果。
3. 遍历原字符串 `s` 中的字符 `ch` ，分情况讨论。当
   - `ch` 是非引号且非下划线，则直接延长进 `lst` 中的最后一个字符串 `lst[-1]` 即可
   - `ch` 是引号，需要将 `ch` 延长进 `lst[-1]` ，同时修改 `flag = not flag` ，因为接下来的**字符是否位于一对引号中的状态切换了**。
   - `ch` 是下划线，则还需要判断当前 `flag` 的值。若
     - `flag = True` ，说明此时的下划线位于一对引号中，不作为分割符使用。下划线的行为和普通字母或数字没有区别，直接延长进 `lst[-1]`
     - `flag = False` ，说明此时下划线作为分割符使用。此时 `lst` 中的最后一个字符串将不再延长，后续的字符串会另起一个新的字符串，因此我们在 `lst` 末尾加入一个新的空串 `""` ，来储存后续的字符。
4. 遍历结束后，由于原字符 `s` 中可能存在一些连续的下划线 `_` ， `lst` 中可能会存在一些空串，我们需要对这些空串进行去重操作。直接使用 `ans = [item for item in lst if item != ""]` 即可完成。
5. 最终，我们选择答案列表 `ans` 中的第 `K` 个元素，将其修改为 `******` 后并输出。注意此处还需要判断ans的长度，如果小于等于 `K` 的话，说明索引 `K` 越界，输出 `ERROR` 。

代码

Python

```
```

```python
# 题目：2024E-敏感字段加密
# 分值：100
# 作者：许老师-闭着眼睛学数理化
# 算法：模拟
# 代码看不懂的地方，请直接在群上提问


# 输入待替换的字段的索引
idx = int(input())
# 输入原字符串s
s = input()

# 初始化列表lst，用于储存根据"_"进行分割的字符串
# 初始化lst包含一个空字符串，用于字符串的延长操作
lst = [""]
# 初始化一个标志Flag，用来判断当前字符ch是否在一对双引号内
Flag = False

# 遍历s中所有的字符ch
for ch in s:
    # 如果遇到一个双引号
    if ch == '"':
        # 对lst中最后一个字符串进行延长操作
        lst[-1] += ch
        # 同时修改Flag的值，表示一对双引号的开始或结束
        Flag = not Flag
    # 如果遇到分割符"_"
    elif ch == "_":
        # 如果此时Flag为True，表示在一对双引号内
        # 不应该进行分割，直接对lst中最后一个字符串进行延长操作即可
        if Flag == True:
            lst[-1] += ch
        # 如果此时Flag为False，表示不一对双引号内
        # 需要开辟一个新的字符串进行延长，故在lst末尾加上一个空串
        else:
            lst.append("")
    # 对于其他字符，直接在lst中最后一个字符串进行延长操作即可
    else:
        lst[-1] += ch


# 由于命令字前后可能出现多个连续的"_"，所以在上述遍历过程中，
# 可能会产生一些无用的空字符串，需要将这些空字符删去，构成列表ans
ans = [item for item in lst if item != ""]
# 如果ans的长度小于等于要求替换的索引idx，则无法完成替换，输出异常
if len(ans) <= idx:
    print("ERROR")
```

```python
# 否则可以进行替换，将索引为idx的字符串替换为六个"*"，再用join()方法进行合并后输出
else:
    ans[idx] = "******"
    print("_".join(ans))
```

Java

```java
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // 输入待替换的字段的索引
        int idx = scanner.nextInt();
        scanner.nextLine(); // 消耗换行符

        // 输入原字符串s
        String s = scanner.nextLine();

        // 初始化列表lst，用于储存根据"_"进行分割的字符串
        // 初始化lst包含一个空字符串，用于字符串的延长操作
        ArrayList<String> lst = new ArrayList<>();
        lst.add("");

        // 初始化一个标志Flag，用来判断当前字符ch是否在一对双引号内
        boolean flag = false;

        // 遍历s中所有的字符ch
        for (char ch : s.toCharArray()) {
            // 如果遇到一个双引号
            if (ch == '"') {
                // 对lst中最后一个字符串进行延长操作
                lst.set(lst.size() - 1, lst.get(lst.size() - 1) + ch);
                // 同时修改Flag的值，表示一对双引号的开始或结束
                flag = !flag;
            }
            // 如果遇到分割符"_"
            else if (ch == '_') {
                // 如果此时Flag为True，表示在一对双引号内
                // 不应该进行分割，直接对lst中最后一个字符串进行延长操作即可
                if (flag) {
```

```
                    lst.set(lst.size() - 1, lst.get(lst.size() - 1) + ch);
                }
                // 如果此时Flag为False，表示不一对双引号内
                // 需要开辟一个新的字符串进行延长，故在lst末尾加上一个空串
                else {
                    lst.add("");
                }
            }
            // 对于其他字符，直接在lst中最后一个字符串进行延长操作即可
            else {
                lst.set(lst.size() - 1, lst.get(lst.size() - 1) + ch);
            }
        }

        // 由于命令字前后可能出现多个连续的"_"，所以在上述遍历过程中，
        // 可能会产生一些无用的空字符串，需要将这些空字符删去，构成列表ans
        ArrayList<String> ans = new ArrayList<>();
        for (String item : lst) {
            if (!item.isEmpty()) {
                ans.add(item);
            }
        }

        // 如果ans的长度小于等于要求替换的索引idx，则无法完成替换，输出异常
        if (ans.size() <= idx) {
            System.out.println("ERROR");
        }
        // 否则可以进行替换，将索引为idx的字符串替换为六个"*"，再用join()方法进行合并后输出
        else {
            ans.set(idx, "******");
            System.out.println(String.join("_", ans));
        }
    }
}

```

C++

```
#include <iostream>
#include <vector>
#include <string>

```

```cpp
using namespace std;

int main() {
    // 输入待替换的字段的索引
    int idx;
    cin >> idx;
    cin.ignore(); // 消耗换行符

    // 输入原字符串s
    string s;
    cin >> s;

    // 初始化列表lst，用于储存根据"_"进行分割的字符串
    // 初始化lst包含一个空字符串，用于字符串的延长操作
    vector<string> lst(1, "");

    // 初始化一个标志Flag，用来判断当前字符ch是否在一对双引号内
    bool flag = false;

    // 遍历s中所有的字符ch
    for (char ch : s) {
        // 如果遇到一个双引号
        if (ch == '"') {
            // 对lst中最后一个字符串进行延长操作
            lst.back() += ch;
            // 同时修改Flag的值，表示一对双引号的开始或结束
            flag = !flag;
        }
        // 如果遇到分割符"_"
        else if (ch == '_') {
            // 如果此时Flag为True，表示在一对双引号内
            // 不应该进行分割，直接对lst中最后一个字符串进行延长操作即可
            if (flag) {
                lst.back() += ch;
            }
            // 如果此时Flag为False，表示不一对双引号内
            // 需要开辟一个新的字符串进行延长，故在lst末尾加上一个空串
            else {
                lst.emplace_back("");
            }
        }
        // 对于其他字符，直接在lst中最后一个字符串进行延长操作即可
        else {
            lst.back() += ch;
        }
    }

```

```cpp
    // 由于命令字前后可能出现多个连续的"_"，所以在上述遍历过程中，
    // 可能会产生一些无用的空字符串，需要将这些空字符删去，构成列表ans
    vector<string> ans;
    for (const string& item : lst) {
        // cout << item << endl;
        if (!item.empty()) {
            ans.push_back(item);
        }
    }

    // 如果ans的长度小于等于要求替换的索引idx，则无法完成替换，输出异常
    if (ans.size() <= idx) {
        cout << "ERROR" << endl;
    }
    // 否则可以进行替换，将索引为idx的字符串替换为六个"*"，再用join()方法进行合并后输出
    else {
        ans[idx] = "******";
        cout << ans[0];
        for (int i = 1; i < ans.size(); ++i) {
            cout << "_" << ans[i];
        }
        cout << endl;
    }

    return 0;
}

```

时空复杂度

时间复杂度： `O(N)` 。需要遍历字符串 `s` 中的每一个字符 `ch` 。

空间复杂度： `O(N)` 。 `lst` 所占空间。