/ 3 4 5 3.

Programming

You are given 2 questions, and need to solve both of them.

Note: You may use the Internet for API search, but communication with other people in any matter is strictly prohibited. Violation to this will be considered as academic misconduct.

Instructions to submit your solutions:

1. The files `QE_prob1.py` (or `QE_prob1.c`) and `QE_prob2.py` (or `QE_prob2.c`) contain your solution to problems 1 and 2, respectively.
2. Remove any debugging or logging code before you submit. It may disturb the automatic grading process, and as a result, you will likely get a lower score.
3. Compress the two files `QE_prob1.py` and `QE_prob2.py` to a single submission `file 20XX_XXXXX.zip` (`20XX_XXXXX` is your SNU student id, *e.g.*, 2022_12345.zip). The submission file should contain at most two files: `QE_prob1.py` and `QE_prob2.py`.
4. Send the submission file to `gsds_qe@aces.snu.ac.kr` from your SNU email account (if it is not an SNU email account, we will not accept your solution). The title of the submission email should be `[QE] 20XX-XXXXX` (*e.g.*, `[QE] 2022-12345`).
5. Make sure that the attached file is easily downloadable from the email message. We will not accept any submission that requires third-party tools or storages (*e.g.*, Google Drive).

길이관여.
adjacent X

1. [50 pts] Consider a string `s` that contains only lower-case alphabets. The number of characters in s does not exceed 10.

(a) [25 pts] Implement a function `foo(s)` that returns a stringtin which no two adjacent characters are adjacent in `s`, and `t` contains exactly the same number of characters as that of `s`. Moreover, when a character appears in `s`, `t` should contain it as often as in `s`. When there is no such string `t`, it returns an empty string "". If there are multiple strings that satisfy the conditions, return any of them. For example, For example,

- When the string `s` is "abcde",`foo(s)` returns "adbec".
- When the string `s` is "abc",`foo(s) returns` "".
- When the string `s` is "",`foo(s) returns` "". ✓✓
- When the string `s` is "abccde",`foo(s)` returns "cacebd".
- When the string `s` is "abcdcef",`foo(s)` returns "cacfbed".

차이점

(b) [25 pts] Implement a function `bar(s)` that returns a string `t` in which no two adjacent characters are adjacent in `s`, and `s` and `t` are different strings. When a character appears in `s`, `t` should contain it only once. When there is no such string `t`, it returns an empty string "". If there are multiple strings that satisfy the conditions, return any of them. For example,

- When the string `s` is "abcde", `bar(s)` returns "adbec".
- When the string `s` is "abc", `bar(s)` returns "".
- When the string `s` is "", `bar(s)` returns "".
- When the string `s` is "abccde", `bar(s)` returns "caebd".
- When the string `s` is "abcdcef", `bar(s)` returns "cafbed".

When you use Python, use the following function signatures:

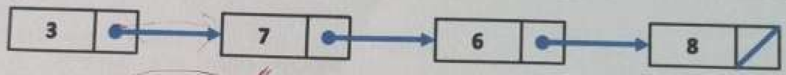
```
def foo(s: str) -> str:
    # your implementation

def bar(s: str) -> str:
    # your implementation
```

When you use C, make sure that the functions return a null-terminated string written on a newly allocated memory(i.e., use malloc). Use the following function signatures:

```
char* foo(const char* s) {
    // your implementation
}
char* bar(const char* s) {
    // your implementation
}
```

You may not use any high-level built-in functions or library routines that directly solve the problems. Your submission file `QE_prob1.py` (or `QE_prob1.c`) should only contain the implementations of `foo(s)`, `bar(s)`, and some auxiliary functions. **You will likely get a lower score if any print or debugging code is in your submission.**

2. [50 pts] Suppose we represent a sequence of integers from 0 to 9 with a list, and it is implemented with a linked list. For example, an integer list [3, 7, 6, 8] is implemented with a linked list:



A list of integers is a palindrome if it is identical to its reversion. Given a list `s` of integers, you will finally implement a function `max_palindromes(s)` that prints a list of sub-lists of `s` that are maximal palindromes. That is, `t` is a list that contains palindrome sub-lists of `s` that are not a sub-list of another palindrome sub-list of `s`. For example, when `s` is [8, 1, 2, 3, 3, 2, 1, 4, 9, 4, 5, 6, 7, 6, 5, 4, 1], `max_palindromes(s)` prints:

$$[[8],\ [1,2,3,3,2,1],\ [4,9,4],\ [4,5,6,7,6,5,4]].$$

When `s` is [8, 1, 2, 3, 3, 2, 1, 0, 4, 9, 1, 2, 3, 3, 2, 1, 9, 1], `max_palindromes(s)` prints:

$$[[8],\ [0],\ [4],\ [9,1,2,3,3,2,1,9],\ [1,9,1]].$$

Assume that the length of `s` is at most 32.

When you use Python, use the following data structure for the linked list node:

```
#node definition
class Node:
    def __init__(self, data, next):
        self.data = data
        self.next = next
```

Otherwise (i.e., you use C), use the following node data structure:

```
// node definition //
struct Node:
{
int data;
    struct node *next;
};
```

(a) [10 pts] Write a function `print_list(s)` that prints the sequence (i.e., the linked list) `s` in the list format. For example, linked list in Figure 3 should be printed as [3, 7, 6, 8].

[illegible]

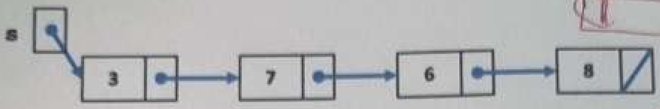
(b) [10 pts] Write a function `palindrome(s)` that returns 1 if the list s is a palindrome. Otherwise, it returns 0.

(c) [10 pts] Write a function `sub_list(s, t)` that returns 1 if the list t is a sub-list of the list s. Otherwise, it returns 0.

↳ list   [ ]   [   ]

(d) [20 pts] Write the function `max_palindromes(s)` that uses `palindrome(s)`, `sub_list(s, t)`, and `print_list(s)`.

In `print_list` and `max_palindromes`, the printed string should only contain square brackets ([and]), commas, digits (0 9), and spaces between them. We accept any formats and orders that describe the answer. For example, "[[1 , 2 , 1] , [3]]" and "[[3],[1,2,1]]" are both OK if the answer is [[1, 2, 1], [3]].



↑

Figure 3: Example for 2(a).

When you use Python, use the following function signatures:

900

```
def print_list(s: Node) -> None:
    # your implementation
def palindrome(s: Node) -> int:
    # your implementation
def sub_list(s: Node, t: Node) -> int:
    # your implementation
def max_palindromes(s: Node) -> None:
    # your implementation
```

When you use C, make sure that the functions return a null-terminated string written on a newly allocated memory(i.e., use malloc). Use the following function signatures:

```
void print_list(struct Node* s) {
    // your implementation
}
int palindrome(struct Node* s) {
    // your implementation
}
int sub_list(struct Node* s, struct Node* t) {
    // your implementation
}
void max_palindromes(struct Node* s) {
    // your implementation
}
```

You may not use any high-level built-in functions or library routines that directly solve the problems. You need to directly handle the node data structures given to solve the problems. i.e. You should not convert the linked list back to a Python list to solve the problem. Your submission file `QE_prob2.py` (or `QE_prob2.c`) should only contain the implementations of `print_list(s)`, `palindrome(s)`, `sub_list(s, t)`, `max_palindromes(s)`, and some auxiliary functions. **You will likely get a lower score if any print or debugging code is in your submission.**