Coding Assignment 1: CS2233

Kindly adhere to the following instructions.

- Please write a C program corresponding to each problem. Your code should be well commented and variable names should be appropriately chosen. Also prepare a `readme` text file where you can mention instructions to run the program/how to take input etc.
- Create a folder and put all the code files and `readme` text file in it, give name to the folder as "yourRollNo", zip the folder and submit it to the google classroom portal.
- Strictly follow the input and output format for each problem.
- Any code that does not follow the input-output criteria won't be evaluated and will get **ZERO**.
- Your code will also be checked against plagiarism (both from web and peer).
- Any form of plagiarism (web/chatGPT/with peers) will be severely penalised and will result in F grade.
- The submission (strict) timeline is 7th September, Thursday, 11 AM.
- Each question consists of 10 marks.

---

1. Given an array of integers. Create a singly linked list along with its head pointer, the problem is to group all the nodes with odd indices, followed by the nodes with even indices, and return the reordered list. The relative order inside the even and odd groups should remain in the input. The following figure illustrates the scenario: Your algorithm's time and space complexity should be $O(n)$ and $O(1)$, respectively

   Input format

   - First line will contain k, which indicates the number of testcases.
   - Following k lines will contain integers of each k arrays.

   Output format

- Your output also should contain k lines, which indicates the output of corresponding k arrays.

Example:
Input:

```
2
2, 1, 3, 5, 6, 4, 7
8, 6, 4, 7, 3, 1, 9, 2, 5
```

Output:

```
2, 3, 6, 7, 1, 5, 4
8, 4, 3, 9, 5, 6, 7, 1, 2
```

Instructions: First, write a method to create the linked list. Then, write a function that takes the head pointer of the linked list as input and performs the task. Your function can access only the linked list via the head pointer only. (15 Marks)

2. Given an array of integers. Create a singly linked list along with its head pointer, return true if it is a palindrome or false otherwise. Your algorithm's time and space complexity should be $O(n)$ and $O(1)$, respectively:

Input format

- First line will contain k, which indicates the number of test cases.
- Following k lines will contain stream of integers.

Output format

- Your output also should contain k lines, which indicates the output of corresponding k arrays.
- Output should be a string, if linked list is palindrome print "True" else print "False".

Example:
Input:

```
2
1, 6, 3, 2, 2, 3, 6, 1
10, 1, 6, 1, 4
```

Output:

```
True
False
```

Instructions: Same as Question 1. (15 Marks)

3. Write a program that takes an infix arithmetic expression as input and outputs the corresponding postfix expression. You can assume that the infix statement is given in the correct format. For example, if the input is `a+b`, then the output should be `ab+`. Your algorithm's time and space complexity should be $O(n)$ and $O(n)$, respectively:

   Input format

   - First line will contain k, which indicates the number of testcases.
   - Following k lines will contain a string of infix statement.
   - Infix string will contain

     $$[a-z] \cup [(,)] \cup [+,-,*,/,\wedge]$$

   - **Note:** Order of precedence of operator is

     $$\{\wedge\} > \{/,*\} > \{+,-\}$$

   Output format

   - Your output also should contain k lines having postfix string corresponding to infix string.

   Example:
   Input:

   ```
   3
   a*(b+c)-d
   a+b*c
   a^b*c+d
   ```

   Output:

   ```
   abc+*d-
   abc*+
   ab^c*d+
   ```

   Instructions: The input is a string of characters, and you can assume that the input given is the correct infix format. You need to check which data structure is suitable for the purpose, and for the problem you need to use the functionality of that data structure only. (10 Marks)

4. The dynamic-set operation SET UNION takes two disjoint sets S_1 and S_2 as input, and it returns a set $S = S_1 \cup S_2$ consisting of all the elements of S_1 and S_2. The operation usually destroys the sets S_1 and S_2. Given two arrays of integers, where the first array consists of the element of S_1 and the second array consists of the element of S_2. The problem is using a suitable list data structure, show how to support SET UNION in $O(1)$ time.

   Input format

- First line will contain k, which indicates the number of test cases.
- Following $2k$ lines will contain integers of each $2k$ arrays.
- The First two lines contain the elements of S_1 and S_2, respectively, for the first test case.
- Next two lines contain the elements of S_1 and S_2, respectively, for the second test case and so on.

Output format

- Your output also should contain k lines, which indicates the output of corresponding k inputs.
- In case of the output set being null, print NULL.

Example:
Input:

```
2
2, 1, 3
5, 6, 4, 7
8, 6, 4, 7, 3, 1
9, 2, 5
```

Output:

```
1, 3, 2, 5, 6, 4, 7
6, 4, 7, 3, 1, 8, 9, 2, 5
```

Instructions: In this problem for simplicity you can assume that your set is the set of integers. A pair of sets S_1 and S_2 are said to be disjoint if they don't have any element in common. You need to check which data structure is suitable for the purpose, and for the problem you need to use the functionality of that data structure only. For e.g. if you plan to use stack data structure for this problem, then you can assess its elements via Push and Pop methods only. You are not allowed to use a doubly-linked list. If you are using a singly linked list, you have access to only the head pointer of the list. (10 Marks)

5. The dynamic-set operation SET INTERSECTION takes two overlapping sets S_1 and S_2 as input, and it returns a set $S = S_1 \cap S_2$ consisting of all the elements of S_1 and S_2. The operation usually destroys the sets S_1 and S_2. Entries of S_1 and S_2 can be in any order. Given two arrays of integers, where the first array consists of the element of S_1 and the second array consists of the element of S_2. The problem is using a suitable list data structure gives the fastest possible implementation of SET INTERSECTION.

Input format

- First line will contain k, which indicates the number of test cases.
- Following $2k$ lines will contain integers of each $2k$ arrays.
- The First two lines contain the elements of S_1 and S_2, respectively, for the first test case.
- Next two lines contain the elements of S_1 and S_2, respectively, for the second test case and so on.

Output format

- Your output also should contain k lines, which indicates the output of corresponding k inputs.
- In case of the output set being null, print NULL.

Example:
Input:

```
2
2, 1, 3
5, 6, 4, 7
8, 6, 4, 9, 5
9, 2, 5
```

Output:

```
NULL
9,5
```

Instructions: In this problem for simplicity you can assume that your set is the set of integers. A pair of sets S_1 and S_2 are said to be overlapping if they have some element in common. You need to check which data structure is suitable for the purpose, and for the problem you need to use the functionality of that data structure only. You are not allowed to use a doubly-linked list. If you are using a singly linked list, you have access to only the head pointer of the list. (10 Marks)

6. Write a C program to implement doubly linked lists using only one pointer value `x->np` per item instead of the usual two `x->next` and `x->prev`. Assume that all pointer values can be interpreted as k-bit integers, and define `x->np` to be `x->np=x->next XOR x->prev`, the k-bit "exclusive-or" of `x->next` and `x->prev`. (The value `NIL` is represented by 0.) Be sure to describe what information you need to access the head of the list. Show how to implement the `SEARCH`, `INSERT`, and `DELETE` operations on such a list. Also, show how to reverse such a list in $O(1)$ time. (30 Marks)

Input Format:

- First line will contain k, which indicates the number of operations to be performed.

- Next k lines will contain the details of each operation as follows:

  The input starts with an integer indicating the operation to be performed, followed by the necessary values for that operation.
  The operations are defined as follows:
  - **1:** Search for the value `x` in the list.
  - **2:** Insert the value `x` into the list.
  - **3:** Delete the value `x` from the list.
  - **4:** Reverse the list.

Output Format:
It consists of k lines corresponding to each k operations.
The output depends on the operation performed:

- **For Search:**
  - Output: `'Element is present'` if the value is found.
  - Output: `'Element is not present'` if the value is not found.
- **For Insert:**
  - Output: The list after inserting the value, with the most recently inserted element appearing first.
- **For Delete:**
  - Output: The list after deleting the specified value.
  - Output: If the deleting element is not present then print `'Element is not present'`.
- **For Reverse:**
  - Output: The list after reversing it.

Test Cases:
Input:

```
9
2 10    %Insert 10%
2 20
4       %Reverse%
1 20    %Search 20%
1 30
2 40
3 30    %Delete 30%
3 40
4
```

Output:

```
10
20 10
10 20
Element is present
Element is not present
40 10 20
Element is not present
10 20
20 10
```