C Programming Assignment 4

Instructions

1. There are two types of questions in this assignment - the first 3 questions are **Non-OJ** questions and are supposed to be submitted on moodle. The last 2 questions need to be submitted on the **OJ** portal.
2. For the Non-OJ part, you are supposed to submit a zip file name `roll_number.zip`. For example, if your roll number is 2023101002, then you are supposed to submit a zip file name `2023101002.zip`. The zip should contain 3 files named q1.c, q2.c, q3.c and a `README.md` specifying the functions you have written, their purpose and data structures you have used.
3. For the Non-OJ part, strictly **follow the naming conventions of the structs and functions** given in the question. Any violation will result in a penalty. For example, in Question 1 the structs must be named `AccountInfo` and functions must be named - `display` , `lowBalanceAccounts` , `transaction` , `createAccount` and `deleteAccount`

▼ Non-OJ

Question 1

Write a structure `AccountInfo` to store the data of customers in a bank. The data to be stored includes:

- **Account Number** (int)
- **Account Type** (enum) - can be either `savings` or `current`
- **Name** (string : number of characters < 50)
- **Balance** in Account (float)

💡 Note:

- All account numbers are unique
- Enum must be used for AccountType
- Create functions for every operation as mentioned below. Use only structs, linked-lists, enums wherever it's possible.
- The program should stop running on receiving `EXIT` command from the user.
- The commands can be called **in any order** i.e first delete then create
- Mention your assumptions or any other functions you require in the README

In the main function, write code to make an interactive program, accepting the following **6 commands** from the user:

`CREATE AccountType Name Amount`

`DELETE AccountType Name`

`TRANSACTION AccountNumber Amount Code`

`LOWBALANCE`

`DISPLAY`

`EXIT`

Output format for each function is mentioned below.

a) Write a function called `display` , to display the account number, account type, name, and balance of each **existing customer (not deleted)** in the following format (sorted in the **order** of account numbers):

```
Account Number     Account Type        Name        Balance
---------------------------------------------------------------
101                savings             Keval       1500.00
102                current             Pramod        80.00
103                current             Aditya       300.00
```

```
104                 savings             Ayan         700.00
105                 savings             Kriti         50.00
```

💡 - If no accounts exist, print: “No Accounts to display”
- You can pass **any parameters** to this function

- This function must be called when the user types `DISPLAY` in the command line interface(CLI).

b) Write functions `createAccount` and `deleteAccount` to create and delete **accounts**(one person(name) can have two accounts of different types: savings/current)

💡 `createAccount` takes parameters: `accountType, Name, Amount`
→ Add the user to the list of Bank accounts, if the user with same AccountType and Username does not exist and print:

Account Number: num

Account Holder: Name

Account Type: Savings/Current

Balance: Rs “balance”
→ If an entry of the user exists with the same AccountType then print:

“Invalid: User already exists!”

`deleteAccount` takes parameters: `accountType, Name`
If the entry does not exist, print:

“Invalid: User does not exist”

otherwise print:

“Account deleted successfully”

- The account number would be generated by you, and has to be given to the user after he creates the account by typing `CREATE AccountType Name Amount` in the command line interface(CLI).
- The first user whose account is created in the bank should have an **account number of 100**.
- Every next user is given an account number of prev + 1. Here, prev is the previously generated AcNo.
- You need to handle the account numbers lost due to deletion from the Bank.

→ For example, let us say that A/c Nos 100, 101 existed and then user with A/C No 100 was deleted then a new user should be assigned the A/C No 100 rather than 102.

💡 Note: A user can create a savings account even if he already has a current account and vice-versa as well. Such operations should be allowed and taken care of.
- For each accountType the user name is unique

- deleteAccount function called when user types `DELETE AccountType Name` in the CLI.

c) Write a function `lowBalanceAccounts` to print the account number, name and balance of each customer with a balance below Rs. 100 like shown below **in order of account numbers**:

```
Account Number: 102, Name: Pramod, Balance: 80
Account Number: 105, Name: Kriti, Balance: 50
```

- This function is called by typing `LOWBALANCE` in the CLI

d) Write a function `transaction` to perform deposit and withdrawal operations.

The function should take parameters in the format: `accountNumber amount code` (1 for deposit, 0 for withdrawal) and print ***"Updated balance is Rs x"*** (x is updated balance)

💡 - If a customer requests a withdrawal and the balance is insufficient (balance - withdrawal amount < 100), display the message: ***"The balance is insufficient for the specified withdrawal."***

Withdrawal of any amount can happen only if there is at least Rs. 100 in the account after the withdrawal, Eg. if the balance is Rs. 150, a withdrawal of Rs. 51 cannot happen.

- If the account does not exist, print ***"Invalid: User does not exist"***

- This function must be called when user types `TRANSACTION AccountNumber Amount Code` in the CLI.

Question 2

Implement a program to perform operations on n-dimensional complex numbers. An n-dimensional complex number is represented as $a_1 + ia_2 + ja_3 + ka_4 +$ ... up to n terms.

You need to implement the following operations:

1. **Addition (`add`):** Given two n-dimensional complex numbers `c1` and `c2`, return a new complex number `ans` such that $\texttt{ans}_t = \texttt{c1}_t + \texttt{c2}_t$ for all terms `t` in the range {1, 2, 3, ..., n}.
2. **Subtraction (`sub`):** Given two n-dimensional complex numbers `c1` and `c2`, return a new complex number `ans` such that $\texttt{ans}_t = \texttt{c1}_t - \texttt{c2}_t$ for all terms `t` in the range {1, 2, 3, ..., n}.

3. **Dot Product (`dot`):** Given two n-dimensional complex numbers `c1` and `c2`, return a float value representing the dot product of `c1` and `c2`.

   $ans = \sum_{t=1}^{n} a_t * b_t$

4. **Cosine Similarity (`cosineSimilarity`):** Given two n-dimensional complex numbers `c1` and `c2`, return a float value representing the cosine similarity between `c1` and `c2`.

$$ans = DOT(a,b)/(|a| * |b|)$$

> 💡 |a| is the mod function and we expect you to implement it too to fetch full marks for `cosineSimilarity` function. Writing another function for it or implementing it in `cosineSimilarity` function only is your choice.

Prompt the user for the operation and dimension `n` and the values of the complex number in the next line. After performing the selected operation, the result should be displayed.

```
ADD n
C1
C2
> Result: <result>

SUB n
C1
C2
> Result: <result>

COSINE n
C1
C2
> Result: <result>

DOT n
C1
C2
> Result: <result>

-1
<exit>
```

Input:

`ADD 4`

`1.5 2.1 3 4.3`

`9.2 7 6 5`

`-1`

Output:

`Result: 10.7 9.1 9 9.3`

💡 - `n` would be provided in the first line of input as the number of dimensions in the complex no.
- The program should **terminate when a -1** is given as input, otherwise the program should keep asking for input from the user for the dimension.
- All outputs displayed should be correct **upto 2 decimal places**.

- You need to strictly **use structs** for this question. Any direct usage of arrays would be given a straight 0
- **Arrays can can be used as an element of a struct for storing an n dimensional complex number**
- Mention your assumptions or any other functions you require in the README

Question 3

Anna, a skilled designer, is working on a project involving a palette of colors. She wants to ensure that no color is repeated, as she believes each color should have its unique presence. To achieve this, she needs to modify the list of colors. The list contains various shades, and she wants to retain each shade exactly once while minimizing the number of nodes in the list. Can you help her achieve this?

Write a program that prompts the user to enter `n` the number of colors in the list, followed by the integers denoting the colors. The program will then **create a linked list** with these colors, **remove any duplicates**, and **return the head** of the modified linked list with unique colors. Finally, it will **print the modified linked list.**

Note: The function `removeDuplicates` should take the original list as a parameter and return the head of the modified linked list → The modification needs to be done INPLACE on the original list, DO NOT just make a new linked list and return.

Constraints:

- The number of nodes in the list `n` is in the range `[0, 300]`.
- `100 <= colors <= 100`
- The list is guaranteed to be **sorted** in non-decreasing order

Sample Input

```
Input:
Enter the number of colors in the list: 5
Enter the colors: 1 2 2 3 4

Output:
Modified Linked List: 1 -> 2 -> 3 -> 4
```

NOTE: OJ Questions are on OJ, getting **AC in all test cases** with **no plagiarism** will fetch you full marks. There are no restrictions about approaches to be followed on OJ.

▼ OJ

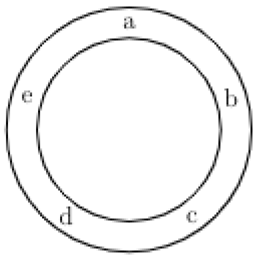
Question 1:

For this question, you've been assigned a challenging mission: embarking on a treasure hunt to locate the illustrious Treasure of the renowned linguistic genius Tapanjali.

Following an arduous search and a long hunt (whose memories have been wiped from your brains, for lore purposes) you eventually reach the final chamber, where the treasure is said to reside.

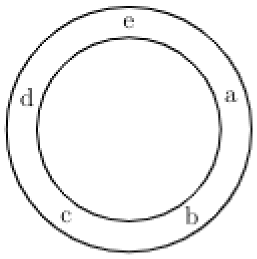
Inside this chamber, you make a curious discovery – two identical rings, each adorned with a matching arrangement of lowercase alphabets. The inscription on these rings is intended to convey a secret message, and it is read by following the clockwise direction from the top.

This ring is read as `abcde`.

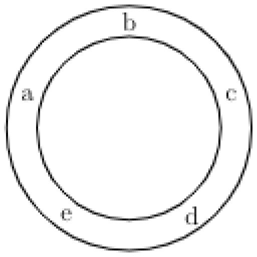


These rings can be rotated such that the secret message they read changes

1. A **clockwise rotation** makes it such that the ring which previously read `abcde` now reads `eabcd`.



2. An **anticlowise rotation** which makes it such that the ring which previously read `abcde` now reads `bcdea`



As you approach the chamber's entrance, you observe two ring-shaped cavities, seemingly designed for inserting these rings. However, your initial attempts at inserting the rings prove futile.

But as you delve further into the writings and engravings of Tapanjali, you uncover a profound insight: Tapanjali had an exceptional fascination with lexicographically smallest and largest strings.

Empowered by this newfound knowledge, you now understand that your task is to rotate the rings in such a way as to **generate** both the **lexicographically smallest** and the **lexicographically largest strings** given a secret message from the ring, thereby unlocking the entrance to the chamber and unveiling the treasure concealed within.

You will be given a non-empty S consisting of lowercase English letters denoting a secret message. Among the secret messages that can be obtained by performing any number of clockwise or anticlowise rotations, find the lexicographically smallest secret message S_{min} and the lexicographically largest secret message S_{max}.

Input Constraints:

- *S* consists of lowercase English letters.
- The length of S is between 1 and 1000 (inclusive).

Input:

S

Output:

Print two lines. The first line should contain S_{min}, and the second line should contain S_{max}..Here, S_{min} and S_{max} are respectively the lexicographically smallest and largest secret messages obtained by performing zero or more clockwise rotations and zero or more anticlockwise rotations on S

Sample Input 1:

```
aaba
```

Sample Output 1:

```
aaab
baaa
```

Sample Input 2:

```
z
```

Sample Output 2:

```
z
z
```

Sample Input 3:

```
abracadabra
```

Sample Output 3:

```
aabracadabr
racadabraab
```

Question 2:

Its the year 2123 and UG1 students are still struggling with their CPro assignment.
They have print some strings in the blazing fast new language called "Z".
Sadly the creators of this new language hate commas, so in order to create an array of strings, you need to separate them with `*` instead of `,`. (so `“hello” * “world”` instead of `“hello”,“world”`)
Your friend from the future asked you for help in his assignment and you being an *expert* in “C” gave him code where you separate the strings with commas. You realised your mistake
and need to immediately fix your code and send it over to save your friend from getting a compilation error (punishable by death).
The strings themselves can have have commas but any comma outside a string needs to be replaced by an `*`. (so `“hello,world" * "23"` is valid but `"hello","world23"` isn’t)
A string is defined to be any sequence of characters enclosed in two double-quotes `"`.
It is guaranteed that there are even number of double-quotes.
More formally, You are given a string *S* of length *N* consisting of lowercase English letters, `,`, and `"`. Let $2K$ be the number of `"` in *S*. For each $i = 1, 2, \ldots, K$, the characters from the

Its the year 2123 and UG1 students are still struggling with their cpro assignment.
They have print some strings in the blazing fast new language called "Z".
Sadly the creators of this new language hate commas, so inorder to create an array of strings, you need to separate them with `*` instead of `,`.
Your friend from the future asked you for help in his assignment and you gave him code where you separate the strings with commas. You realised your mistake
and need to immediately fix your code and send it over to save your friend from getting a compilation error (punishable by death).
The strings themselves can have have commas but any comma outside a string needs to be replaced by a *.
A string is defined to be any sequence of characters enclosed in two double-quotes `"`.
It is guaranteed that there are even number of double-quotes
More formally, Given an string of length N with 2k double-quotes, all commas between (2i-1)th and (2i)th double-quotes i belong to [1, k] stay commas, and any

other comma needs to be replaced with asterisk(`*`). The given string only contains lowercase english letters, commas(`,`) and double-quotes(`"`)

$(2i-1)^{th}$ `"` through the $(2i)^{th}$ `"` are said to be **enclosed**.

Your task is to replace each `,` in *S* that is **not** an enclosed character with `*` and print the resulting string.

Input Constraints:

- *N* is an integer between 1 and 2×10^5, inclusive.
- *S* is a string of length N consisting of lowercase English letters, `,`, and `"`.
- *S* contains an even number of `"`.

Input:

N

S

Sample Input 1:

```
8
"a,b"c,d
```

Sample Output 1:

```
"a,b"c*d
```

Sample Input 2:

```
5
,,,,,
```

Sample Output 2:

```
*****
```

Sample Input 3:

```
20
a,"t,"c,"o,"d,"e,"r,
```

Sample Output 3:

```
a*"t,"c*"o,"d*"e,"r*
```