Abstract Data Types (ADT's)

Assignment 1
Data Structures and Algorithms
Due date: 10 March, 2021

1.1 Complex Number ADT: The goal is to define and implement an ADT for n-dimensional complex numbers. A n-dimensional complex number is of the form $a_1 + ia_2 + ja_3 + ka_4 + ...$ upto n terms.

The code must consist of three files as follows :
1. *complex.h* - Which defines the ADT using the header files.
2. *complex.c* - Which implements the functions defined later.
3. *main.c* - Which must use the ADT and perform various operations on input and produce output as specified later.

The functions to be implemented are as follows :
1. Addition: (ADD) - $a_t + b_t \ \forall t \in \{1, 2, ..., n\}$
2. Subtract (SUB) - $a_t - b_t \ \forall t \in \{1, 2, ..., n\}$
3. Mod (MOD) - $\sqrt{\sum_{t=1}^{n} a_t^2}$
4. Dot product (DOT) - $\sum_{t=1}^{n} a_t b_t$

Input
The first line contains a string TYPE and an integer N, denoting the type of the operation that you are supposed to make and the dimension of the complex numbers respectively.
If the TYPE is MOD, then the next line contains N integers, denoting the complex number.
Else, the next 2 lines contain N integers each, denoting the complex numbers on which you are supposed to make the operations.

Output
If the TYPE is DOT, then print a single integer denoting the output of the Dot Product operation.
If the TYPE is MOD, then print a single floating point denoting the output of the Mod operation, rounded off to exactly 2 decimal places.
Else, print N integers, denoting the final complex number you get after doing the operations.

Constraints
$1 \leq N \leq 10^6$
$-10^5 \leq$ Each of the N integers $\leq 10^5$

Sample Test Cases

Input	Output
ADD 4 1 2 3 4 9 7 6 5	10 9 9 9
MOD 4 2 0 1 2	3.00

Note
There is no automated evaluation for this problem. TAs will evaluate each code manually.

1.2 Music Player ADT: Create a *musicplayer.h* that will contain the basic functionalities of a music player. You DO NOT have to implement the functions, just state them.

Note
This question is open-ended on purpose and you are free to design the music player as you want. Also note that one ADT can contain another ADT. Again, there is no automated evaluation and codes will be evaluated manually.

Linked Lists

Assignment 1
Data Structures and Algorithms
Due date: 10 March, 2021

2.1 Singly Linked List:

For the code for linked list discussed in class (Find it here), extend the functionality and add these functions:

1. **FindLast(L,X)** - Find the last occurence of element X in linked list and return its position (assume 0 indexed). Return -1 if there is no such element.
2. **DeleteAll(L,X)** - Delete all elements of value X from the linked list. (Do nothing if there is no such element)

Sample Test Case

Assume the current state of linked list to be $1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4 \longrightarrow 2$ then -
FindLast(L,2) should return 4 and DeleteAll(L,2) should make the linked list $1 \longrightarrow 3 \longrightarrow 4$.

Note
There is no automated evaluation for this problem. TAs will evaluate each code manually.

2.2 Polynomial ADT:

You are supposed to extend the Polynomial ADT discussed in class (Find it here) and implement the following functions:

1. **AddPolynomials(Poly A, Poly B, Poly C)** - Add two Polynomials A and B and store the result in Polynomial C
2. **SubtractPolynomials(Poly A, Poly B, Poly C)** - Subtract Polynomial B from Polynomial A and store the result in Polynomial C
3. **DeleteTermByExponent(Poly P, unsigned long int exponent)** - Delete the term with the given exponent from Polynomial P. (Do nothing, if it already does not exist)
4. **GetMiddle(Poly P)** - Get the pointer to ceil(n/2)-th term of Polynomial P containing n terms, without storing n as part of the Polynomial ADT.
5. **DeleteAllTerms(Poly P)** - Delete all terms (make all coefficients equal to 0)
6. **DeletePoly(Poly P)** - Delete the polynomial (remove its existence)
7. **PrintPoly(Poly P)** - Print the polynomial in standard form. If all elements are zero, just print 0 and if it had been deleted, print "Polynomial does not exist".
8. OPTIONAL: **GetQuartile(Poly P, int x)**: Write a single function which can return pointer to the x^{th} quartile of the polynomial, without storing n as a part of the Polynomial ADT. GetMiddle corresponds to the 50^{th} quartile, 75^{th} quartile will be $ceil((3*n)/4)$ and so on...

Sample Test Cases

Assume that in the current state, we have the following two polynomials:

1. Poly A: $5x^4 + 4x^3 + 2x + 7$
2. Poly B: $-7.2x^3 + 4x^2 - 4$

Input	Output
AddTerm A 7 -3.2	Done
AddTerm B 6 3	Done
PrintPoly A	$-3.2x^7 + 5x^4 + 4x^3 + 2x + 7$
PrintPoly B	$3x^6 - 7.2x^3 + 4x^2 - 4$
PrintPoly P	Polynomial does not exist
AddPolynomials A B C	Done
PrintPoly C	$-3.2x^7 + 3x^6 + 5x^4 - 3.2x^3 + 4x^2 + 2x + 3$
SubtractPolynomials C B D	Done
PrintPoly D	$-3.2x^7 + 5x^4 + 4x^3 + 2x + 7$
DeletePoly C	Done
PrintPoly C	Polynomial does not exist
DeleteAllTerms D	Done
DeleteTermByExponent A 0	Done
PrintPoly A	$-3.2x^7 + 5x^4 + 4x^3 + 2x$
PrintPoly D	0

Note
There is no automated evaluation for this problem. TAs will evaluate each code manually.

Stacks

Assignment 1
Data Structures and Algorithms
Due date: 10 March, 2021

3. Stacks:

3.1: Write a header file *my_stack.h*. Define a data type *my_stack* which builds the stack **using linked list**.

3.2: Define interface for functions to be defined on *my_stack* in the *my_stack.h* header file:

1. **push**: given an element, push it on the stack
2. **pop**: pop the top element from the stack

3.3: Write the routines for the above two functions in *my_stack.c*.

3.4: Write a C program which illustrates the above operations by building a good command line interface, i.e. ask the user which operations he/she would like to do in a stack of 2-D Complex Numbers. (Elements of the stack contain 2 values x, y representing complex number $x + iy$)

1. push
2. pop
3. exit

After operations 1 and 2 always display the stack content.

OPTIONAL: How can you modify the above program for n-dimensional complex numbers?

Note
There is no automated evaluation for all the problems. TAs will evaluate each code manually.

Submission Instructions
You need to submit this assignment on the Courses Portal. Your submission is expected to be a <RollNumber>.zip file.

NOTE: Strict actions would be taken against anyone found involved in any kind of plagiarism either from the internet or from other students.