CIS 1201 Finals Mock Exam – Test 2 of 2 (70 points, 120 minutes)

Instruction: Write the code of the functions based on the given specifications. Make your code as efficient as you can and avoid redundant statements when possible. Only one return statement shall be allowed for every function created (except when recursion is applied). Conditions in the iteration and alternation statements must be relational operators.

Place your answers at the designated areas of the answer sheet provided.

Problem A (15 points)

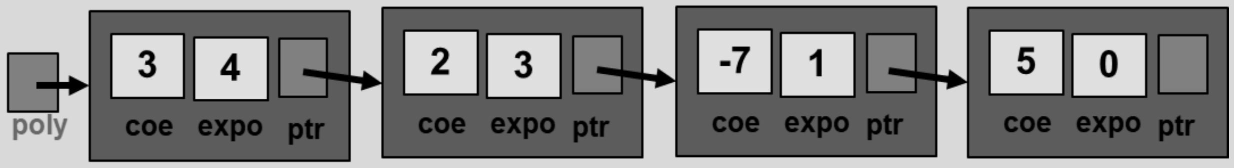
Given the list below and the value of x as parameters, create a function which returns to the calling function the integer result of the polynomial expression contained in the linked list.

Restrictions: No existing functions can be called, except for the function itself.
Note: Full 15 points will be awarded if recursion is applied AND implemented correctly in your code; otherwise, highest possible score is only 12 points.

Data structure definition:

```
typedef struct term{
   int coe, expo;
   struct term* ptr;
}*polynomial;
```

Sample diagram:



Example: If x = 2 then
$f(x) = 3x^4 + 2x^3 - 7x + 5$
$f(x) = 55$

Problem B (15 points)

Given a structure containing two strings of the same length (max 30 characters - consisting of all capital letters), create a function that checks if it is possible to recreate the second-string input using the characters from the first string.

If it is possible, the extras string will be remained empty, and the function will return 1.

Otherwise, the extras string will store the list of characters that are present in the second string but not at the first (in alphabetical order), and the function will return 0.

Restrictions: No string functions allowed. Can only create a maximum of one auxiliary data structure.

Data structure definition:	

`#define MAX 30`
 `typedef struct{`
 `  char str1[MAX];`
 `  char str2[MAX];`
 `  char extras[MAX];`
 `}twostrings;`
	Example 1:
	S1: "SKIDOODLE"
	S2: "SKIDDADLE"
	Extras: "AD"

	Example 2: S1: "TOMMARVOLORIDDLE" S2: "IAMLORDVOLDEMORT" Extras: ""
	Example 3: S1: "IDONTKNOW" S2: "HGFEDCBAA" Extras: "AABCDEFGH"

Problem C (15 points)

Given a list of credit card numbers stored in a 2D array (every row contains an array of 16 integer elements where each integer represents a digit) and the number of 16-digit credit cards (which will be expected to be at around 7,000 to 10,000 cards, to be processed by a bank for account inspection purposes following a serious case), create a function that will check the validity of each credit card number. The function shall return an array of the status of the credit cards processed - for every card checked, a character 'V' will be appended to the array if the credit card is valid and 'I' if otherwise. Put a -1 at the end of the array to be returned. Contents inside the array can be manipulated.

The validity of these cards can be checked by first doubling every other digit starting from the first digit. If the doubled digit results in a 2-digit number, the sum of the digits shall then be recorded. The digits obtained shall be added afterwards. The card number is valid only if the resulting sum is a multiple of 10.

Example in checking a credit card:

Given: 5234 8213 3410 1298

Digits	5	2	3	4	8	2	1	3	3	4	1	0	1	2	9	8
Doubled	10	2	6	4	16	2	2	3	6	4	2	0	2	2	18	8
Digit sum	1	2	6	4	7	2	2	3	6	4	2	0	2	2	9	8

Sum of digit sums = 60 → Valid since the sum obtained is a multiple of 10.

Problem D (25 points)

A convenience store is making adjustments to their product display every end of the year by selling new products and removing all the products under certain brands which are less frequently purchased by the customers. Below is the definition of the structure of product items in a certain convenience store stored using array implementation.

```
#define SIZE 0xFA
typedef struct{
   int year;
   int month;
   int day;
}date;

typedef struct{
   char brand[20];
   char name[20];
   char servingSize[16];
}ProdName;
```

```
typedef enum{
   PERISHABLE,
   NONPERISHABLE
}expiration;

typedef union{
   date expDate;
   char comment[4]; /* "N/A" */
}expInfo;
```

```
typedef struct{
   ProdName PName;
   float price;
   expiration exp;
   expInfo info;
}products;

typedef struct{
   products prod[SIZE];
   int ctr; /*holds actual number
              of products*/
}productList;
```

Create the following functions:

a) Given a list of products and the name of the file containing the new lists of products, create a function that will read from the list of new products one record at a time and append only the non-perishable products found in the file to the end of the given array list.

b) Given a list of products and the brand name of the products to be removed, create a function that will open an existing store file named "removedProducts.dat" and append to the end of the file all the products containing the brand name. The product records written to the file will be removed as well from the array list. Also, return the number of products removed.

Restrictions: No auxiliary aggregate variables (structure, array, etc.) allowed for all functions.
Only primitive variables can be declared.

-- END OF TEST 2 --

-- REVIEW YOUR ANSWERS!! --

Problem reference/credits (all items are modified from the original):
Problem A – CIS 1201 Final Exam (2019) Test 3B
Problem B – DSC-USC x CIT Programming Competition (2021) Preliminary Problem 5 of 6
Problem C – CIS 1101/1201 problem submission (2020) by Wayne Dayata
Problem D – ADT List Exercise 2 (2021) by Marvie Gasataya

NOTE: PLEASE REVIEW ADT LINKED LIST IMPLEMENTATION – SEE PREFINAL MOCK EXAM