HW5

(deadline 2019/11/26)

手寫題:

3. Imagine we implement a list using a dummy node at the beginning of the list. The dummy node does not carry any data. It is not the first data node, it is an empty node. Figure 5-27 shows a list with a dummy node.

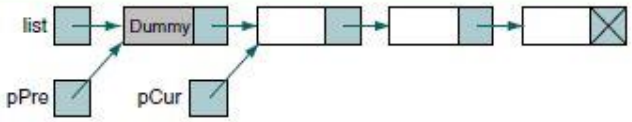

FIGURE 5-27 Linked List Implementation for Exercise 3

Write the code to delete the first node (the node after the dummy node) in the list.

4. Write the code to delete a node in the middle of a list implemented as a linked list with the dummy node (see Exercise 3). Compare your answer with the answer to Exercise 3. Are they the same? What do you conclude? Does the dummy node simplify the operation on a list? How?

程式題:

26. Write a program to read a list of students from a file and create a list. The program should use a linked list for implementation. Each node in the linked list should have the student's name, a pointer to the next student, and a pointer to a linked list of scores. There may be up to four scores for each student. The structure is shown in Figure 5-33.

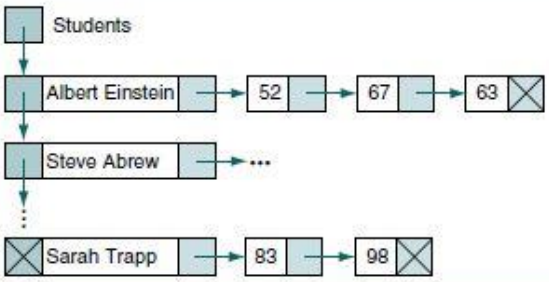

FIGURE 5-33 Data Structure for Project 26

The program should initialize the student list by reading the students' names from the text file and creating null score lists. It should then loop through the list, prompting the user to enter the scores for each student. The scores' prompt should include the name of the student.

After all scores have been entered, the program should print the scores for each student along with the score total and the average score. The average should include only those scores present.

The data for each student are shown in Table 5-3.

Student name	Score 1	Score 2	Score 3	Score 4
Albert Einstein	52	67	63	
Steve Abrew	90	86	90	93
David Nagasake	100	85	93	89
Mike Black	81	87	81	85
Andrew Dijkstra	90	82	95	87
Joanne Nguyen	84	80	95	91
Chris Walljasper	86	100	96	89
Fred Albert	70	68		
Dennis Dudley	74	79	77	81
Leo Rice	95			
Fred Flintstone	73	81	78	74
Frances Dupre	82	76	79	
Dave Light	89	76	91	83
Hua Tran	91	81	87	94
Sarah Trapp	83	98	94	93

TABLE 5-3 Data for Project 26

29. Write a program that adds and subtracts polynomials. Each polynomial should be represented as a list with linked list implementation. The first node in the list represents the first term in the polynomial, the second node represents the second term, and so forth.

Each node contains three fields. The first field is the term's coefficient. The second field is the term's power, and the third field is a pointer to the next term. For example, consider the polynomials shown in Figure 5-34. The first term in the first polynomial has a coefficient of 5 and an exponent of 4, which is then interpreted as $5x^4$.

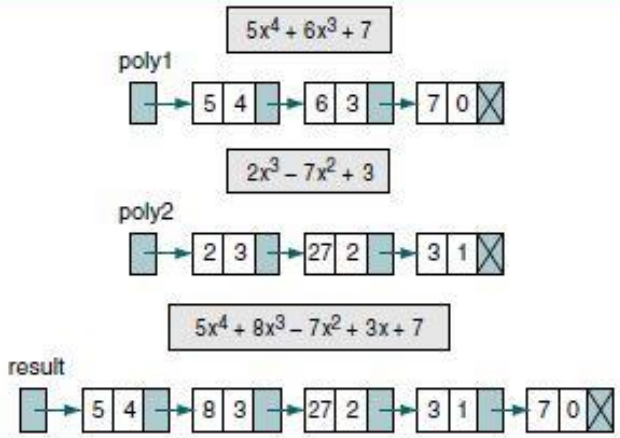

FIGURE 5-34 Example of Polynomials for Project 29

The rules for the addition of polynomials are as follows:

a. If the powers are equal, the coefficients are algebraically added.
b. If the powers are unequal, the term with the higher power is inserted in the new polynomial.
c. If the exponent is 0, it represents x^0, which is 1. The value of the term is therefore the value of the coefficient.
d. If the result of adding the coefficients results in 0, the term is dropped (0 times anything is 0).

A polynomial is represented by a series of lines, each of which has two integers. The first integer represents the coefficient; the second integer represents the exponent. Thus, the first polynomial in Figure 5-35 is

```
5   4
6   3
7   0
```

To add two polynomials, the program reads the coefficients and exponents for each polynomial and places them into a linked list. The input can be read from separate files or entered from the keyboard with appropriate user prompts. After the polynomials have been stored, they are added and the results are placed in a third linked list.

The polynomials are added using an operational merge process. An operational merge combines the two lists while performing one or more operations—in our case, addition. To add we take one term from each of the polynomials and compare the exponents. If the two exponents are equal, the coefficients are added to create a new coefficient. If the new coefficient is 0, the term is dropped; if it is not 0, it is appended to the linked list for the resulting polynomial. If one of the exponents is larger than the other, the corresponding term is immediately placed into the new linked list, and the term with the smaller exponent is held to be compared with the next term from the other list. If one list ends before the other, the rest of the longer list is simply appended to the list for the new polynomial.

Print the two input polynomials and their sum by traversing the linked lists and displaying them as sets of numbers. Be sure to label each polynomial.

Test your program with the two polynomials shown in Table 5-4.

Polynomial 1		Polynomial 2	
Coefficient	Exponent	Coefficient	Exponent
7	9	–7	9
2	6	2	8
3	5	–5	7
4	4	2	4
2	3	2	3
6	2	9	2
6	0	–7	1

TABLE 5-4 Text Data for Project 29

30. In older personal computers, the largest integer is 32,767 and the largest long integer is 2,147,483,647. Some applications, such as cryptography and security algorithms, may require an unbounded integer. One way to store and manipulate integers of unlimited size is by using a linked list. Each digit is stored in a node of the list. For example, Figure 5-35 shows how we could store a five-digit number in a list.

Although the list in Figure 5-35 is represented as moving from right to left, there is no physical direction in a list. We represent it in this way to clarify the problem.

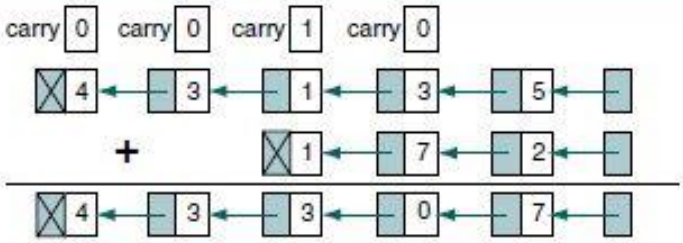

FIGURE 5-35 Integer Stored in a List for Project 30

To add two numbers, we simply add the corresponding digit in the same location in their respective lists with the carry from the previous addition. With each addition, if the sum is greater than 10, we need to subtract 10 and set the carry to 1. Otherwise, the carry is set to 0.

Write an algorithm to add two integer lists. Design your solution so that the same logic adds the first numbers (units position) as well as the rest of the number. In other words, do not have special one-time logic for adding the units position.

36. You have been assigned to a programming team writing a system that will maintain a doubly linked, multilinked list containing census data about cities in the United States. The first logical list maintains data in sequence by the 2000 population data. The second logical list maintains the same data using the 1990 census data. Your assignment is to write the insert routine(s). To completely test your program, you also need to write routines to print the contents of the lists.

The structure for the data is shown here:

```
Logical definition:
Metropolitan area                50 characters
Population - 2000                   integer
Population - 1900                   integer
```

The input is a file containing the census data in Table 5-5.

Metropolitan area	Census population	
	2000	1990
New York–No. NJ	21,199,865	19,549,649
Los Angeles area	16,373,645	14,531,529
Chicago area	9,157,540	8,239,820
Washington–Baltimore	7,608,070	6,727,050
San Francisco area	7,039,362	6,253,311
Philadelphia–Atlantic City area	6,188,463	5,892,937
Boston area	5,819,100	5,455,403
Detroit area	5,456,428	5,187,171
Dallas–Fort Worth	5,221,801	4,037,282
Houston–Galveston area	4,669,571	3,731,131
Atlanta area	4,112,198	2,959,950
Miami–Fort Lauderdale	3,876,380	3,192,582
Seattle area	3,554,760	2,970,328
Phoenix area	3,251,876	2,238,480
Minneapolis–St. Paul	2,968,806	2,538,834
Cleveland area	2,945,831	2,859,644
San Diego area	2,813,833	2,498,016
St. Louis area	2,603,607	2,492,525
Denver area	2,581,506	1,980,140
San Juan, PR, area	2,450,292	2,270,808

TABLE 5-5 Twenty Largest Metropolitan Areas in the United States

To verify your program, print the list in sequence, both forward and backward, for both census years. To conserve paper print the data two-up, as shown in Table 5-6.

Census Data for 1990	Population	Census Data for 2000	Population
Metropolitan area		Metropolitan area	
01 San Juan, PR, area	2450292	01 San Juan, PR, area	2270808
02 Denver area	2581506	02 Denver area	1980140
...		...	
19 Los Angeles area	16373645	19 Los Angeles	14531529
20 New York-No. NJ	21199865	20 New York-No. NJ	19549649
20 New York-No. NJ	21199865	20 New York-No. NJ	19549649
19 Los angeles Area	16373645	19 Los Angeles	14531529
...		...	
02 Denver area	2581506	02 Denver area	1980140
01 San Juan, PR, area	2450292	01 San Juan, PR, area	2270808

TABLE 5-6 Example of PrintOut

39. Write a program that builds an array of linked lists (do not use the ADT). The data for each linked list are to be read from a set of existing files and inserted into the list. The program is to accept a variable number of files that are determined at run time by asking the user for the number of files and their names. The data structure is shown in Figure 5-38.

After the program builds the lists, print them to verify that they are complete. Test the program with the following four files:

a. 150 110 130 100 140
b. 200 280 240 220 260
c. 390 300 330 360
d. 480 440 400

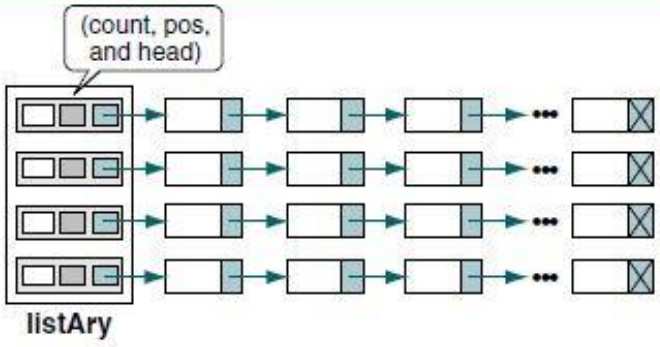

FIGURE 5-38 Structure for Array of Linked Lists