CCU CSIE

Data Structure Course - 2017 Fall

Computer-Based Test I Questions

Instructor: Yu-Ling Hsueh

TA: Nick, Kile, Ruby, and Jay

Note: You should pay attention to the newline characters in the outputs.

1. Implement sparse matrix transpose and multiplication

Given two sparse matrices A and B below, please implement sparse matrix multiplication to compute A*B using the **fast transpose** approach introduced in the lecture. The input consists of the first matrix followed by the second matrix and the first line for each matrix contains the number of rows and columns of the matrix. You must transpose matrix B before multiplying it by A. First, output the result of the starting position of each column in a sequence of numbers when performing the fast transpose. Subsequently, output the multiplication result for non-zero terms only with the first line to represent the number of rows, columns, and non-zero terms.

Note: You must define a triple struct <row, col, value> and use an 1-D array to store non-zero term in the matrix.

$$\underbrace{\begin{bmatrix} 0 & 3 & 0 & 1 \\ 0 & 1 & 3 & 4 \\ 0 & 0 & -2 & 0 \end{bmatrix}}_{A} \times \underbrace{\begin{bmatrix} 3 & 1 \\ -1 & 0 \\ 5 & 0 \\ 0 & 2 \end{bmatrix}}_{B} = \begin{bmatrix} -3 & 2 \\ 14 & 8 \\ -10 & 0 \end{bmatrix}$$

Test Case

Please test your program with Input, and then check the answers with Output.

.

Listing 1: Implement sparse matrix multiplication.

```
1  Input：
2  3 4
3  0 3 0 1
4  0 1 3 4
```

```
0 0 -2 0
4 2
3 1
-1 0
5 0
0 2

Output：
Starting position for each column of B:
0 3
A * B:
3 2 5
0 0 -3
0 1 2
1 0 14
1 1 8
2 0 -10
```

2. KMP algorithm

Please implement the failure function of the KMP algorithm to preprocess a given pattern P, and use this algorithm to compare the string T and pattern P, which are stored in a separate 1-D array (index starts with 0). You should print out the result of failure function F first. If the pattern P is found in string T, you should output (Yes,x,y), where x and y represent the starting and ending char address in String T. If patter P cannot be found in string T, you should output (Not found).

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 2 : KMP algorithm

```
Input：
T : A A B Z A B Z A B C Z
P : A B Z A B C

Output：
F : -1 -1 -1 0 1 -1
Yes,4,9
```

3. Invert a singly linked list

Please implement a function to invert a singly linked list in C. The input includes two lines: a sequence of input numbers, and a sequence of partition codes. The number of the input numbers is multiplication by 9 (e.g, 9, 18, 27,…), so that we can divide the numbers into three partitions. We define 1 as the front partition, 2 as the middle partition, and 3 as the rear partition. As the example shown in the following input, there are 9 numbers so each partition has three numbers (3 4 8) (9 11 22) (13 99 23). You must read each partition code sequentially and invert the partition. After that, insert the inverted partition into the back of the sequence of numbers. For example, the first partition code is 1, the front partition (3 4 8) is inverted to (8 4 3) and insert them into the back of the numbers. So the sequence of numbers becomes 9 11 22 13 99 23 8 4 3. Read the next partition code and continue the operation based on the output of the previous the partition-inversion-insertion operation. You must output each list of numbers for each operation.

Note: You must use a linked list; otherwise, no points will be given.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 3 : Invert a singly linked list

```
Input：
3 4 8 9 11 22 13 99 23
1 2 3 3 1 2

Output：
9 11 22 13 99 23 8 4 3
9 11 22 8 4 3 23 99 13
9 11 22 8 4 3 13 99 23
9 11 22 8 4 3 23 99 13
8 4 3 23 99 13 22 11 9
8 4 3 22 11 9 13 99 23
```

4. Implement a stack by using linked lists

Please finish a program which can push and pop a stack by using a linked list in C. The input includes a sequence of push and pop operations. The output includes two lines of the results: (1) the size of the stack, and (2) the status of the stack from the

bottom to the top.

Note: You must use linked list; otherwise, no points will be given.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 4 : Implement a stack by using a linked list

```
1  Input：
2  Push 2
3  Push 6
4  Push 8
5  Pop
6  Push 83
7  Push 33
8  Push 17
9  Pop
10 Pop
11 Push 13
12 Push 25
13 Pop
14
15 Output：
16 4
17 2 6 83 13
```

5. Structure and array

Please implement a structure array <name, day, score, area>, read the input file, and store the test data in an array for statistical analysis. The data in the input file (illustrated in the following table) was collected from night markets and a data tuple consists of name, opening hours, evaluation, and area columns. The maximum number of night markets is 10. After reading the file, according to the following three questions, output the questions and their answers as shown in List 5.

Name	Opening hours	Evaluation ⭐	Area
花園夜市	四、六、日	5	台南市
大東夜市	一、二、五	4	台南市

武聖夜市	三、六	4	台南市
民雄夜市	一	2	嘉義縣
東石夜市	二	3	嘉義縣
士林夜市	每日	3	台北市
公館夜市	每日	3	台北市
師大夜市	每日	4	台北市
逢甲夜市	每日	3	台中市
大甲夜市	四	2	台中市

Note: You must use an array and struct; otherwise, no points will be given.

Test Case

Please test your program with Input and, and then check the answers with Output.

Listing 5: Structure and array

```
1  Input：
2  花園夜市,467,5,台南市
3  大東夜市,125,4,台南市
4  武聖夜市,36,4,台南市
5  民雄夜市,1,2,嘉義縣
6  東石夜市,2,3,嘉義縣
7  士林夜市,1234567,3,台北市
8  公館夜市,123567,3,台北市
9  師大夜市,1234567,4,台北市
10 逢甲夜市,1234567,3,台中市
11 大甲夜市,4,2,台中市
12 Output：
13 評價最高的夜市：花園夜市
14 星期三有開的夜市：武聖夜市、士林夜市、公館夜市、師大夜市、逢甲
   夜市
15 台中的夜市：逢甲夜市、大甲夜市
```

6. Infix to Postfix Conversion

Please finish a program which can change the expression from the infix form to the postfix form in C using a stack. The input operands are lowercase letters「*a*」to「*z*」; meanwhile, the input operators are 「*」,「/」,「+」, and 「-」. Furthermore, the precedence of the operators is「*」=「/」>「+」 =「-」. The output includes two lines of the results: (1) the postfix expression, and (2) the maximum number of elements in

the stack during the computation.

Test Case

Please test your program with Input1 and Input2, and then check the answers with Output1 and Output2.

Listing 6: Infix to Postfix Conversion.

```
Input1：
a + b * c
Output1：
a b c * +
2

Input2：
a / b - c + d * e - a * c
Output2：
a b / c - d e * + a c * -
2
```

7. Postfix expression evaluation

Please finish a program which can perform postfix expression evaluation and count the max top in stack during the process in C. The input is a prefix expression. The operands are lowercase letters「*a*」to「*z*」; the input operators are「*」,「/」,「+」, and「-」. Furthermore, the precedence of the operators is「*」=「/」>「+」=「-」. First, convert the prefix expression to the postfix form. After that, you should use a stack to process postfix form and record the stack size for each step. The output includes two results: (1) the postfix expression, and (2) the stack size for each step.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 7 : Postfix expression evaluation

```
Input：
- + a * - / b c d e / f g

Output：
a b c / d - e * + f g / -
```

```
6 1 2 3 2 3 2 3 2 1 2 3 2 1
```

8. Circular queue

Please implement a circular queue **using an array** in C. Please program the enqueue(), dequeue(), and display() functions. The length of an array is defined as six; as a result, there are five spaces to store data. In the input file, we define 1 as enqueue() with a data separated by a space; 2 as dequeuer(), and 3 as display().

Note: You must use an array to implement a circular queue; otherwise, no points will be given.

Test Case

Please test your program with Input1 and Input2, and then check the answers with Output1 and Output2.

Listing 8: Circular queue

```
1  Input1:
2  3
3  1 5
4  1 10
5  3
6  2
7  1 15
8  1 20
9  3
10 Output1:
11 The queue is empty.
12 5 10
13 10 15 20
14 Input2:
15 1 3
16 1 5
17 1 7
18 1 9
19 1 11
20 3
21 1 13
22 2
```

```
2
3
Output2:
3 5 7 9 11
The queue is full.
7 9 11
```

9. Implement a binary search tree with linked lists

Please implement a binary search tree with linked lists and traversal with in-order, post-order, pre-order. The input consists of a sequence of numbers. You must build a binary search tree according to the input order. The output includes three lines of the results: (1) the in-order traversal, (2) the post-order traversal, and (3) the pre-order traversal.

Note: You must use linked lists; otherwise, no points will be given.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 9 : Implement a binary search tree with linked list

```
Input：
4 6 8 9 12 5 7

Output：
4 5 6 7 8 9 12
5 7 12 9 8 6 4
4 6 5 8 7 9 12
```

10. Insert and delete a number of binary search tree

Implement a binary search tree with insert(x) and delete(x) functions using an array. You must build a binary search tree according to the input order. The input consists of a sequence of numbers in the first line. The insert/delete function and the number to be insert/deleted are shown in the remaining lines. For the delete operation, you must use the node with the largest key from the left sub-tree to replace the deleted node. Finally, print out the entire tree in level order from the top to the down. If there are more than two numbers in the same level, you should print out the numbers from the left to the right

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 10 : Insert and delete a number of binary search tree

```
Input：
4 5 1 0 6 7 2
insert(44)
delete(6)
insert(3)
delete(2)

Output：
4 1 5 0 3 7 44
```

11. Heap sort

Given a sequence of n numbers, build a heap using an array and implement a heap sort that results in a sorted numbers in ascending order. The output includes two lines of the results: (1) the contents of the array that stores the heap in each step, and (2) the sorted numbers in ascending order.

Test Case

Please test your program with Input1 and Input2, and then check the answers with Output1 and Output2.

Listing 11: Heap sort

```
Input1：
1 4 5 0 6 7 2

Output1：
0 1 2 4 6 7 5
1 4 2 5 6 7
2 4 7 5 6
4 5 7 6
5 6 7
6 7
7
0 1 2 4 5 6 7
```

```
Input2：
3 7 5 9 1

Output2：
1 3 5 9 7
3 7 5 9
5 7 9
7 9
9
1 3 5 7 9
```

12. Equivalent Relations

Given *n* equivalent relations, find the equivalent classes. Each line in the input shows the equivalent relation that consists of a pair of numbers separated by a space. Each line in the output is an equivalent class.

Note: You must sort each equivalent class in ascending order, and print the equivalent classes based on the smallest number of each class in ascending order.

Test Case

Please test your program with Input1 and Input2, and then check the answers with Output1 and Output2.

Listing 12: Equivalent Relations

```
Input1：
0 4
3 1
6 10
8 9
7 4
6 8
3 5
2 11
11 0

Output1：
0 2 4 7 11
1 3 5
6 8 9 10
```

```
16
17 Input2：
18 1 3
19 2 5
20 1 7
21 5 6
22 6 0
23
24 Output2：
25 0 2 5 6
26 1 3 7
```

13. Maze

Given a 10 by 10 maze with one entrance and one exit, find the way to the exit (8,8) from the entrance (1,1). There is a border (represented by 1s) around the maze. There are only four moving directions: right > down > left > up. You must use a stack to store the moving path. Input is a 10 by 10 maze with a border. Output must be the moving path that consists of steps separate by ",".

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 13: Maze

```
1  Input：
2  1111111111
3  1011000011
4  1000011111
5  1011111111
6  1000111111
7  1110111111
8  1110111111
9  1110111111
10 1110000001
11 1111111111
12 Output：
13 (1,1),(1,2),(2,2),(3,2),(4,2),(4,1),(5,1),(6,1),(7,1),(6,1),(5,1),(4,1),(4,2),(3,2),
   (2,2),(1,2),(1,3),(1,4),(2,4),(3,4),(3,5),(3,6),(3,7),(3,8),(4,8),(5,8),(6,8),(7,8),
   (8,8)
```

14. Polynomial addition, subtraction, and multiplication

Please use a linked list to implement the polynomial addition, subtraction, multiplication. Your program should read the input file and correctly display a polynomial using linked lists. The format of a polynomial in the input file is shown as follows. Each term consists of a coefficient and an exponent, separated by a space. The first polynomial is listed followed by a ":" and the second polynomial.

$3.5x^8 + 6x^7 + 9x^4 + 2.3x^2 + 1$

Coefficient	exponent
3.5	8
6	7
9	4
2.3	2
1	0

- A list structure example :

```
struct PolynomialNode{
    float coef;
    int expont;
    struct PolynomialNode *link;
};
typedef struct PolynomialNode P_Node;
typedef P_Node *P_Node_P
```

The following functions are needed in the program:

Attach( ) - Insert each term in a polynomial into a linked list by this function.
PrintPoly( ) - Print all the terms in the linked list.
PolyAdd( ) - Implement the polynomial addition and return the result.
PolySub( ) - Implement the polynomial subtraction and return the result.
PolyMul( ) - Implement the polynomial multiplication and return the result.

Output:

A(x): Use Attach( ) and PrintPoly( ) to store and print polynomial 1.
B(x): Use Attach( ) and PrintPoly( ) to store and print polynomial 2.
C(x): Use PolyAdd( ) and PrintPoly( ) to print the result of polynomial 1 + polynomial 2.
D(x): Use PolySub( ) and PrintPoly( ) to print the result of polynomial 1 - polynomial 2.
E(x): Use PolyMul( ) and PrintPoly( ) to print the result of polynomial 1 × polynomial 2.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 14: Polynomial addition, subtraction, and multiplication

```
1 Input：
2 3.5 8
```

```
6.0 7
9.0 4
2.3 2
1.0 0
:
9.0 8
3.4 7
6.0 5
5.0 3
100.0 0
Output：
A(x)=3.5x^8+6.0x^7+9.0x^4+2.3x^2+1.0
B(x)=9.0x^8+3.4x^7+6.0x^5+5.0x^3+100.0
C(x)=12.5x^8+9.4x^7+6.0x^5+9.0x^4+5.0x^3+2.3x^2+101.0
D(x)=-5.5x^8+2.6x^7-6.0x^5+9.0x^4-5.0x^3+2.3x^2-99.0
E(x)=31.5x^16+65.9x^15+20.4x^14+21.0x^13+117.0x^12+48.1x^11+50.7x^
10+61.8x^9+359.0x^8+662.2x^7+17.5x^5+900.0x^4+5.0x^3+230.0x^2
+100.0
```

15. Recursive

Please implement a program which can add up the digits of a number to a sum less than 10. Each input contains a single positive integer *n*. Add each digit of *n* and repeat the process until the sum is ＜ 10. For example, n= 567. First, add up the digits of *n* (i.e., 5+6+7 = 18). Because the sum is larger than 10, you need to keep adding up the digits of 18. Finally, the output is 9 (i.e., 1+8 = 9).

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 15 : Recursive

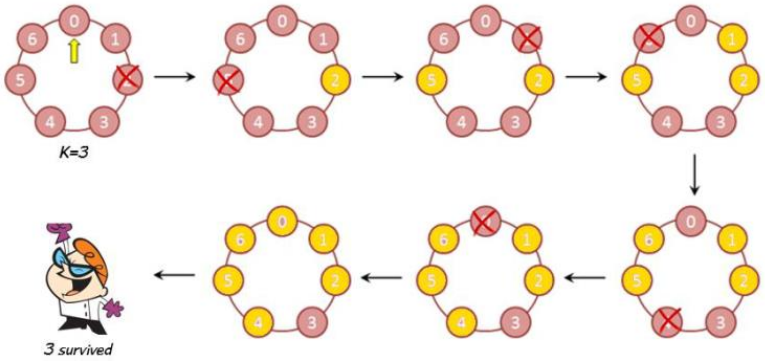
```
Input：
31
69
1234567892

Output：
4
6
```

```
9 | 2
```

16. Circular linked list

Please implement a program which creates a **circular linked list** and displays the elements in the list. Next, use it to solve the Josephus problem. The illustration of the processing about Josephus problem is the following figure. Assume there are eight people. Each number represents a person. *k* is used as a specified number of people to be skipped. Therefore, in the first round, #2 is killed. The procedure is repeated with the remaining people, starting with the next person who is not dead, going in the **same direction** and skipping the same number of people, until only one person remains, and is survived. The default direction is **clockwise**. The input file contains a list of names separated by commas. The second line is the *k* value. Please output the sequence of the people who are killed during the process.



An Illustration of the Josephus Problem.

Note: You must use linked lists; otherwise, no points will be given.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 16: Circular linked list

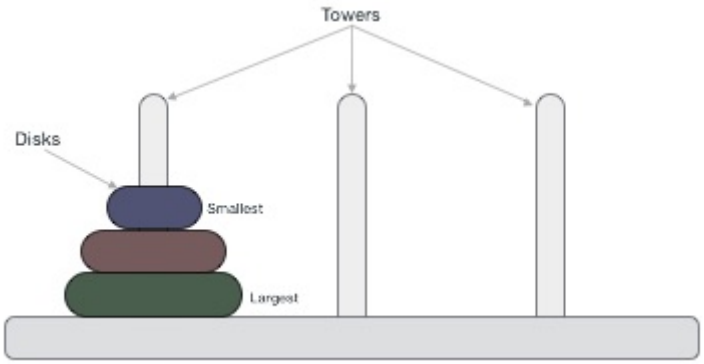
```
1 | Input：
2 | Jay,Tom,Wayne,Kevin,Helen,Jim,Kile,Ruby,Nick,Patty,Mike,Kurt
3 | 2
4 |
5 | Output：
6 | Tom is killed.
7 | Kevin is killed.
8 | Jim is killed.
```

```
Ruby is killed.
Patty is killed.
Kurt is killed.
Wayne is killed.
Kile is killed.
Mike is killed.
Helen is killed.
Jay is killed.
Nick is survived.
```

17. Tower of Hanoi

The **Tower of Hanoi** is a mathematical game or puzzle. It consists of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape. Tower of Hanoi puzzle with *n* disks can be solved in minimum 2^n-1 steps. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1. Only one disk can be moved at a time.
2. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack.
3. No disk may be placed on top of a smaller disk.



The input is a positive integer that represents the number of disks you have, and the output should contain the minimum of steps and its moving processes.

Test Case

Please test your program with Input, and then check the answer with Output.

Listing 17: Tower of Hanoi

```
Input：
```

```
3

Output：
7
Move sheet from A to C
Move sheet from A to B
Move sheet from C to B
Move sheet from A to C
Move sheet from B to A
Move sheet from B to C
Move sheet from A to C
```

18. Use linked list to implement palindrome

Please make a palindrome. (word that reads the same when you reverse it). The input consist of several lines ending with EOF. Each line contains a non-empty string of upper case and lower case English letters (‘A’-‘Z’ and ‘a’-‘z’). The length of the string is less than or equal to 100,000.

Note: You must use linked lists; otherwise, no points will be given.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 18 : Use linked list to implement palindrome

```
Input：
Aawweeaa
Abba
asdfghj

Output：
aawweeaaeewwaa
abbabba
asdfghjhgfdsa
```

19. Palindrome

Given a string with length of *n*, reverse every M letters in the string using a stack. For example, given a string “cardogbug” and if *M* =3, the first three letters are reversed to “rac”, the next three letters are reversed to “god”, and the last three letters are

reversed to “gub”. As a result, the output is “racgodgub”. If M > *n*, output the original string without inversing it.

Note: You must implement it by a stack; otherwise, no points will be given.

Test Case

Please test your program with Input, and then check the answers with Output.

Listing 19 : Palindrome

```
Input ：
10 ASDFASDFA
3 DFJIDOJBIICVJOZXOJQSA
Output ：
ASDFASDFA
JFDODIIBJVCIZOJJOXASQ
```

20. KMP (2)

Please read the given file (input.txt) that involves the NAME and INDEX data. At first, you should initiate the MAIN_STRING as ‘NULL’. You then will use MAIN_STRING to store the read data from the file (input.txt). The value of NAME is the data you should insert to the MAIN_STRING, and the INDEX data represents the index position where you should insert at the next time. For an instance, after the initiation, you read ‘jim’ and ‘1’ from file (input.txt), hence, the MAIN_STRING will be ‘jim’. Then, you keep reading ‘tom’ and ‘2’ from the file(input.txt), you should insert ‘tom’ into the ‘1’ position of the current MAIN_STRING, therefore, the MAIN_STRING will be ‘jtomim’. The last line “P:aaaaa” in the input file is used to implement the KMP algorithm.

After read file, implement the KMP algorithm to preprocess a read MAIN_STRING, and print the LSP[] which is used to skip characters while matching.

Listing 19 : Palindrome

```
Input ：
jim,1
tom,2
patty,3
helen,4
aaaaa,4
P:aaaaa
```

```
Output ：
Final string : jtopatheleaaaaantymim
Last string: jtopathelentymim
Lps = {0,1,2,3,4}
```