#_____   ID _____   Name _____

✧   共 12 題，130 分。**請看清楚題目再作答**。

✧   第 5~12 題請直接回答在試題卷。1~4 請用答案卷(如回答在試題卷，請於答案卷註明)。

✧   題目有難有易 (1-B, 2B 相對較難)，請分配時間，作答不用按照順序。

✧   若依照題目條件限制，有多個答案，請回答其中任一個。如果某題沒有答案，請回答
    "本題無解"。

1.   Please read a recent news:

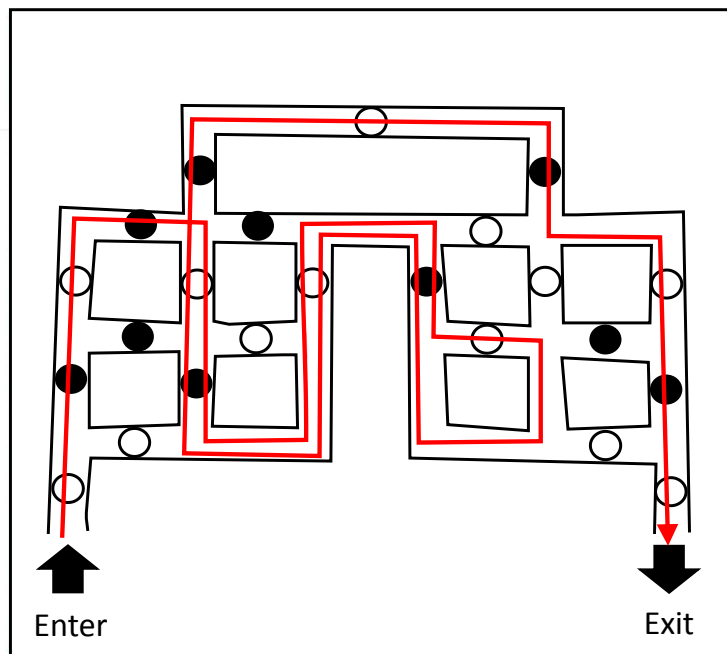(The above red path is the solution of this maze.   For your reference only.   This question does not indeed require you to solve the maze.   Please note that a same place can be visited twice. Therefore, a basic DFS algorithm (even if it takes white and black marks into consideration) cannot solve the maze because a basic DFS visits each place no more than one time.

We would like to develop a program that can solve this type of mazes. Please answer the following questions:

A. [5%] Please show an **adjacency list type** data structure that can represent this type of maze.
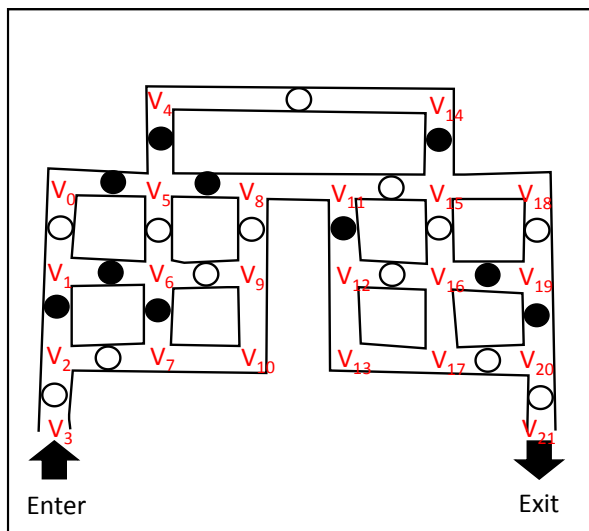
Let each crossroad be a vertex, each road segment be an edge, and each mark be the edge cost. According to this rule, we can have an adjacency list representation:

edge (7, 2) with a White marker
edge (7, 6) with a Black marker
edge (7, 10) with No mark
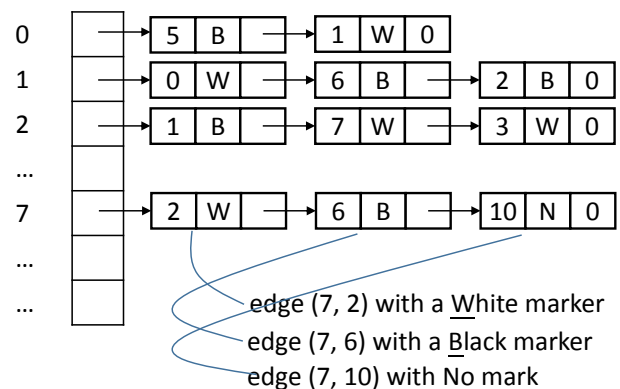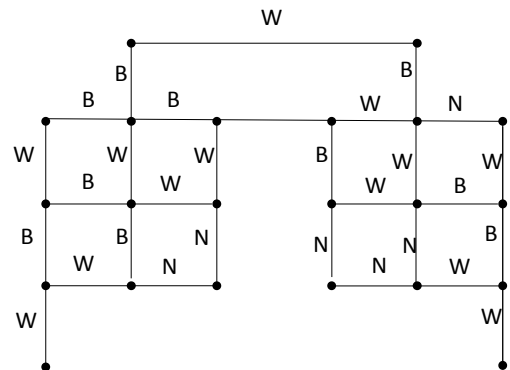
Please note that the following red path is not a valid solution.   Therefore, if we consider white and black marks as vertices when constructing the graph, we can run into this kind of issue.
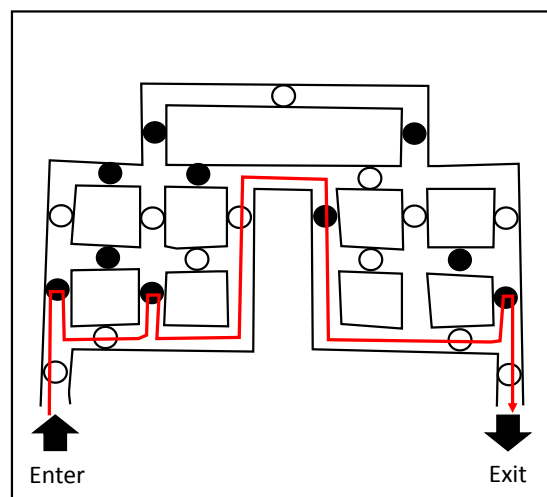
B. [10%] Please describe the algorithm of your program that can solve this type of maze.
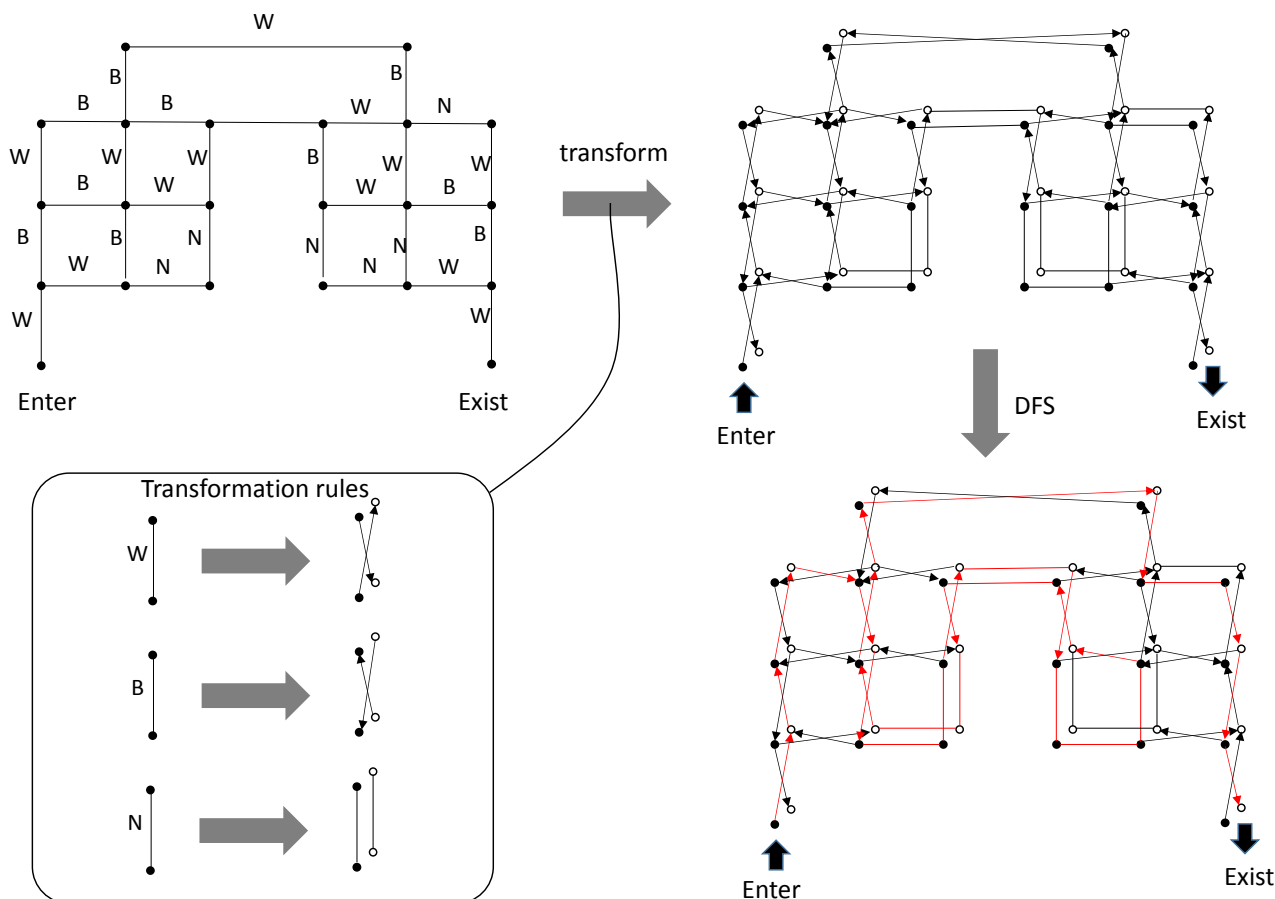
Hint: One recommended strategy is transforming this problem into our known problem. You can describe a method that transforms any maze with black and white marks into an equivalent (等效的/等價的) maze (e.g., a directed graph) without such marks. If finding a path in the latter maze is a known problem, your work is done.

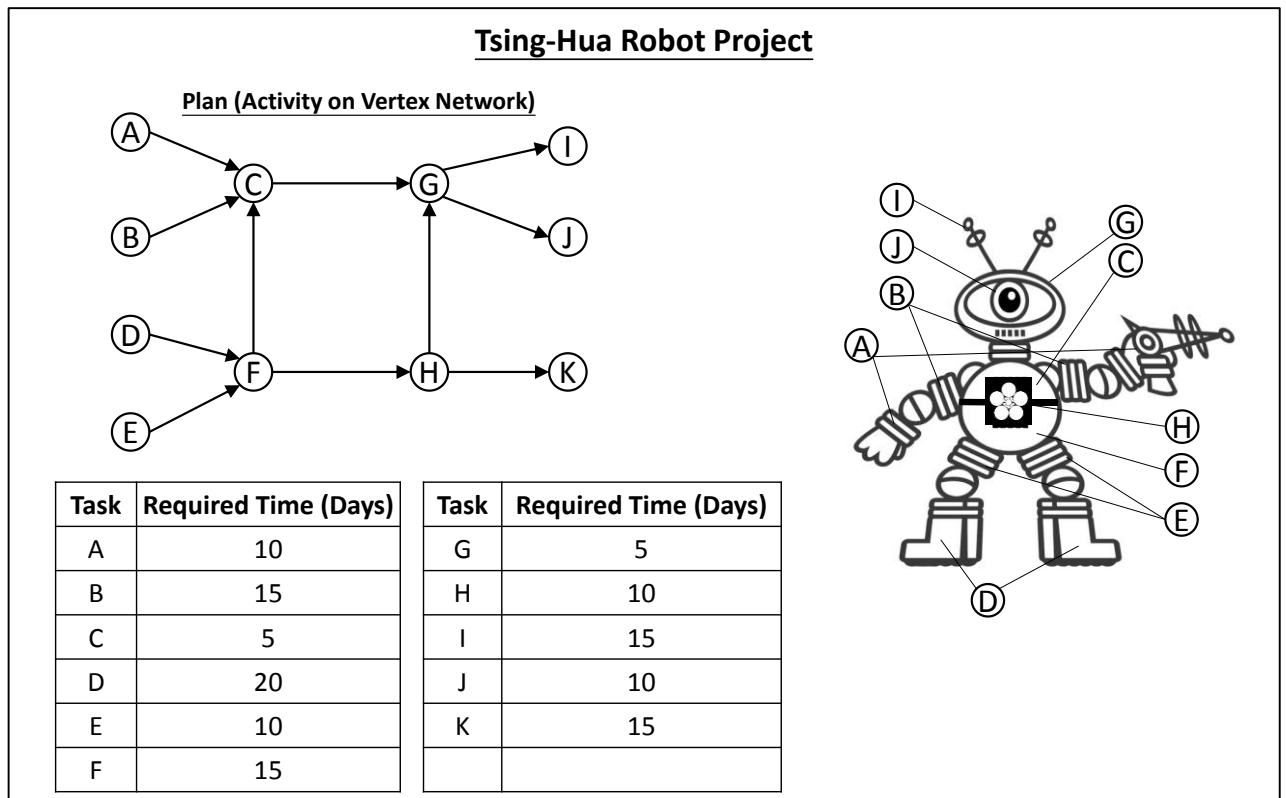A possible algorithm is as follows:

C. [5%] Let **n** and **e** be the number of vertices and edges of the input maze. Please analyze the time complexity of your algorithm.

Transformation: $\theta(n + e)$
DFS: $\theta(n + e)$
Overall: $\theta(n + e)$

2. 校長 would like NTHU students to build a robot for the upcoming NTHU-NCTU competition! The project plan is set up as follows.

Tsing-Hua Robot Project

Plan (Activity on Vertex Network)

Task	Required Time (Days)	Task	Required Time (Days)
A	10	G	5
B	15	H	10
C	5	I	15
D	20	J	10
E	10	K	15
F	15		

A. [5%] Please derive a **topological order** of the above activity on vertex (AoV) network. Please answer the one with **the minimum dictionary order**. For example, if both "ABCD" and "ADBC" are valid topological orders, please answer the former one.

A B D E F C H G I J K

B. [5%] Please convert the above AoV network to an **activity on edge (AoE) network**.
Hint: You need to add a start and a finish vertex. Furthermore, you can add edges with zero cost to handle the case that multiple activities depend on one activity (e.g., C and H both depend on F).
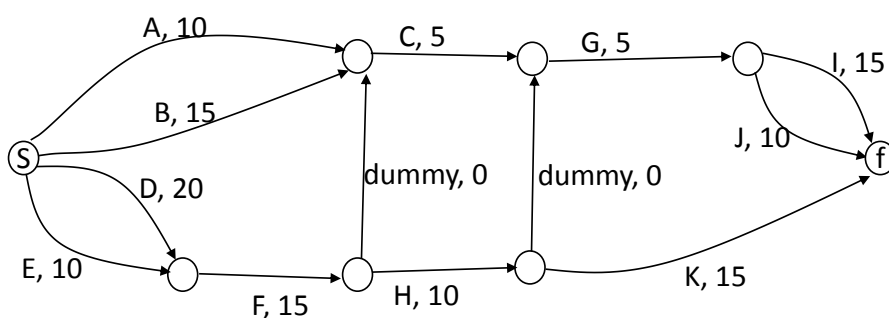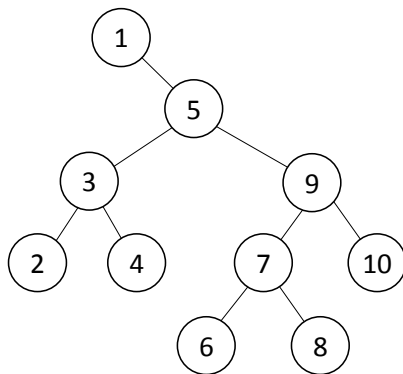
C. [5%] Please derive the **critical activities** and the **earliest finish time** of the project.
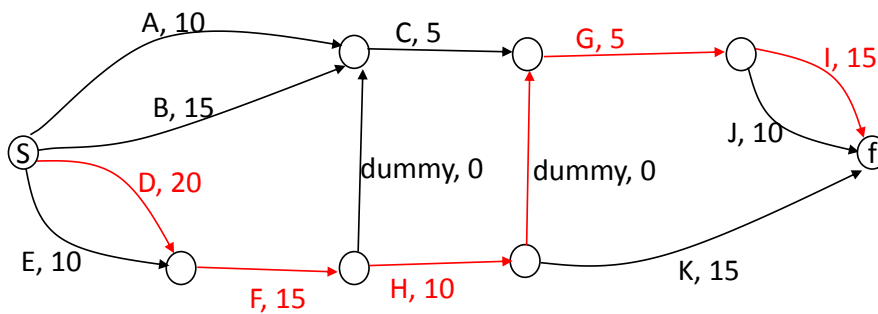

Critical activities: D, F, H, G, I
Earliest finish time = path cost of D, F, H, G, I = 65

3. [6%] Please insert ten keys **1, 5, 9, 7, 3, 2, 6, 4, 8, 10** into an empty **binary search tree (BST)** and plot the final tree.   We want <u>each left child to hold a smaller number than its parent</u> in the BST.

4. [6%] Please plot a 9-node binary tree whose **level-order** sequence is "**D O R M I T O R Y**" and whose **in-order** sequence is "**D I R T Y R O O M**".   (Just kidding! no offense.)
Hint: Specifically, the sequences are "**D O_1 R_1 M I T O_2 R_2 Y**" and "**D I R_1 T Y R_2 O_1 O_2 M**"
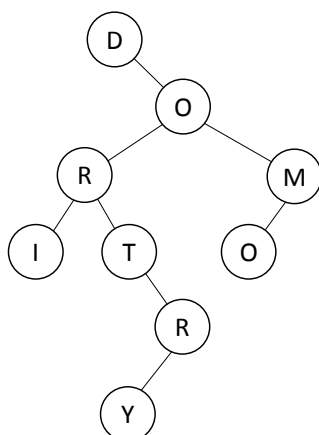
5. A **Bloom filter** uses the following **three hash functions** to index a **10-entry table**.

    h1(k) = k % 10

    h2(k) = (2*k) % 10

    h3(k) = (k*k) % 10

A. [5%] Please show the table after three keys, **3, 5, 8**, are added to the Bloom filter.

    h1(3) = 3, h2(3) = 6, h3(3) = 9 → these positions are set for the key 3

    h1(5) = 5, h2(5) = 0, h3(5) = 5 → these positions are set for the key 5

    h1(8) = 8, h2(8) = 6, h3(8) = 4 → these positions are set for the key 8

    so the table becomes:

| Table index:	0	1	2	3	4	5	6	7	8	9
	1	0	0	1	1	1	1	0	1	1

B. [5%] Given the above condition, please list all the **false-positive** keys from **1 to 9**.

    From 1 to 9, the only false-positive key is 4. h1(4) = 4, h2(4) = 8, h3(4) = 6 → these three positions are all set in the Bloom filter, but 4 should not belong to the Bloom filter. The Bloom filter answers a wrong "yes" (i.e., false positive).

6. [10%] Please complete the following **infix-to-postfix** conversion process.
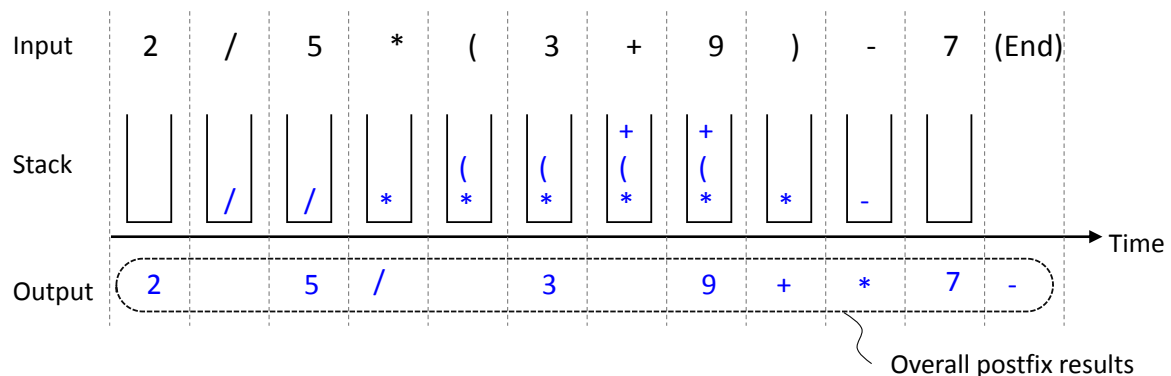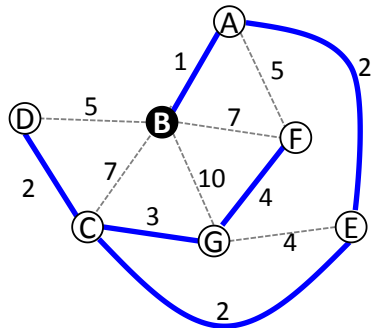

7. [6%] Please add threads (to form a **threaded-binary tree**) to the following binary tree
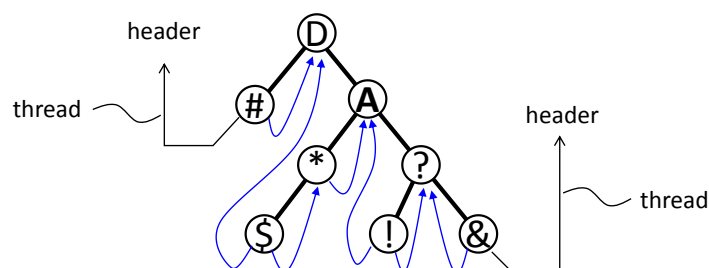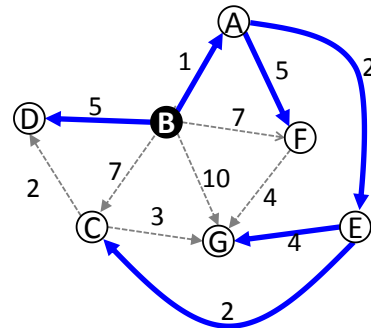
8. [10%] Please plot the **spanning tree** obtained using **Prim's** algorithm and **Dijkstra's** algorithm using B as the **starting vertex**.

Prim's (B)

Dijkstra's (B)

9. Please answer the following questions about a BST and a heap.

A. [4%] Please show the BST after we remove the key "A".

Remove "A" from the BST

or
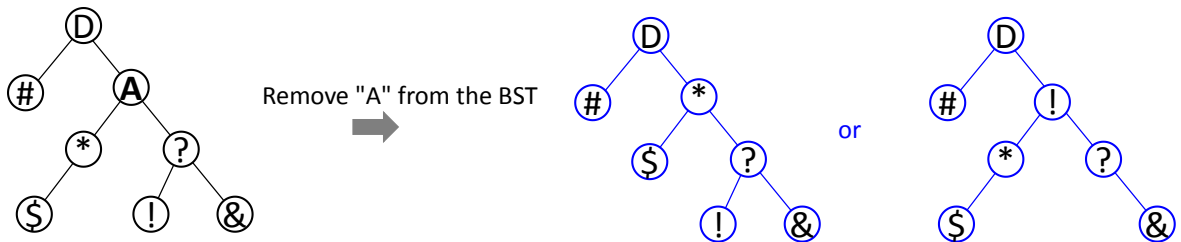
B. [4%] Please list the order among all the eight keys (including A) in the BST. Either ascending or descending order is good (由大至小或由小至大均可)

D $ * A ! ? &   or
& ? ! A * $ D #

C. [4%] Please show the heap after we remove the key, "#". The order among the keys are the same as that of the above BST.

Remove "#" from the heap
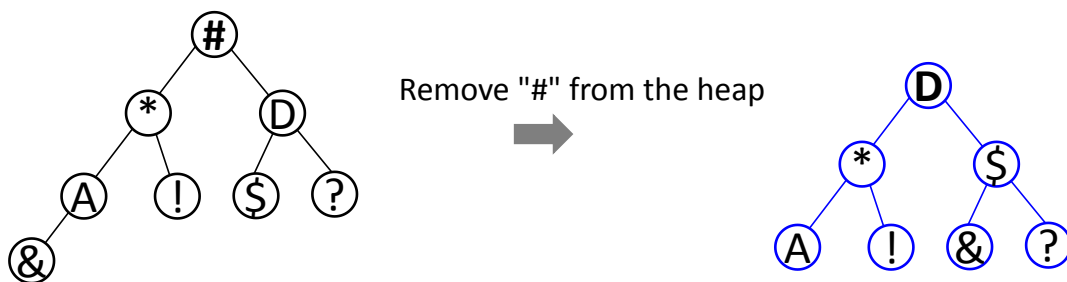
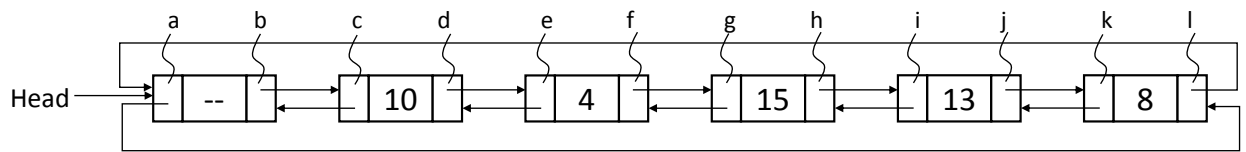10. Please perform the first swapping step of **Quick Sort** on a **doubly linked list with a header**. We want **ascending order** (由小排到大).
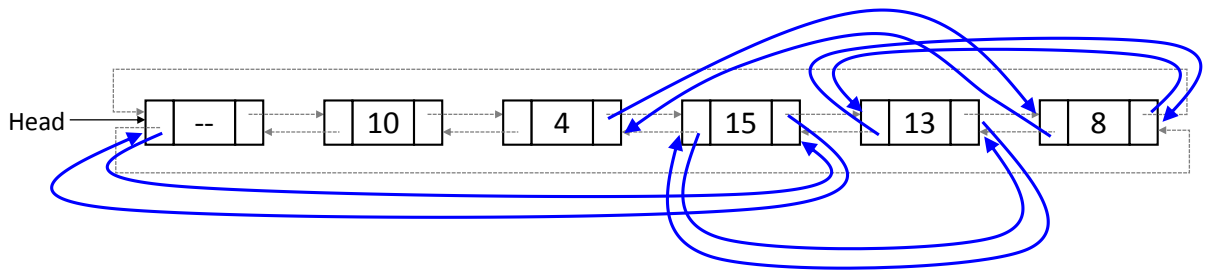
15 and 8 need swapping.

A. [5%] Quick Sort first picks 10 as the pivot, then Quick Sort swaps two keys <u>by change links (instead of only moving keys)</u>.

Please tell which links (a, b, …, k, l) need to be change?   <u>a, f, g, h, i, j, k, l</u>

B. [5%] Please **redraw those changed links** (arrows) to reflect the above swapping step.
(請畫出被改動的 links 的箭頭來反映上述 swapping step):

C. [5%] Please show pseudocode for swapping any two nodes in a doubly linked list with a header (<u>by changing links instead of just moving keys</u>):

```
void swap (NODE * a, NODE * b)
// a and b are pointers (a != b) pointing to the two nodes we want to swap.
{
    if (b->right == a) { swap(a, b); }

    a->left->right = b;      b->right->left = a;

    if (a->right == b) {
       a->right = b->right;    b->left = a->left;
       a->left = b;        b->right = a;
    } else {
       a->right->left = b;    b->left->right = a;
       swap(a->left, b->left);     swap(a->right, b->right);
    }
}
```

11. [10%] Please complete the following **LSD-first Radix Sort** process that sort the following names according to the **dictionary order**.

Sort LSD ⤬⇄   ⤬⇄   ⤬⇄   ⤬⇄   Sort MSD ⤬⇄

Input		MSD	LSD				Output	
"JANE"	J A N E	J A N E	T O M	T O M	J A N E	D A V E	"DAVE"
"JANNY"	J A N N Y	T O M	J A N E	T O M M Y	J A N N Y	D A V I D	"DAVID"
"DAVID"	D A V I D	D A V E	D A V E	J A N E	D A V E	J A N E	"JANE"
"TOM"	T O M	D A V I D	D A V I D	J A N N Y	D A V I D	J A N N Y	"JANNY"
"DAVE"	D A V E	J A N N Y	T O M M Y	D A V E	T O M	T O M	"TOM"
"TOMMY"	T O M M Y	T O M M Y	J A N N Y	D A V I D	T O M M Y	T O M M Y	"TOMMY"
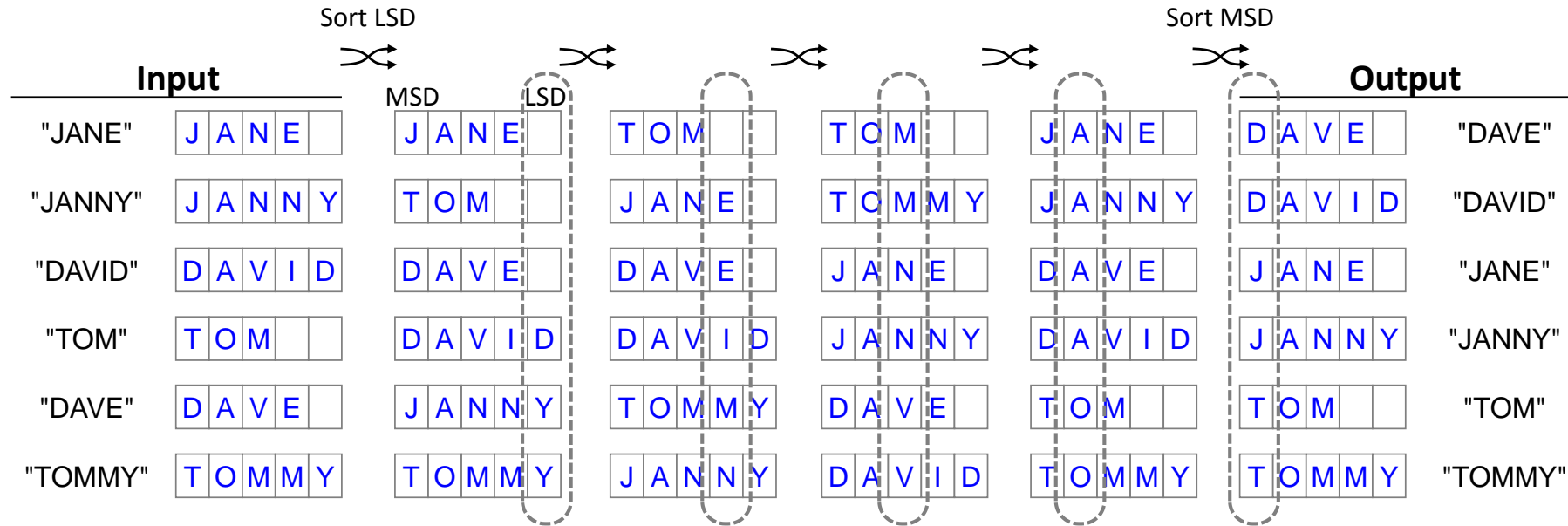
12. [10%] Please show the spanning tree of the following graphs using depth-first search (DFS) and breadth-first search (BFS) traversal given a starting vertex.
    - DFS(A) means vertex A is the starting vertex of DFS.
    - Please prioritize edges with a smaller number during traversal.

DFS (A)   DFS (G)   BFS (B)   BFS (F)