大同大學 102 學年度 轉學入學考試試題

考試科目：資料結構　　　　所別：資訊工程學系　　　　第 1/1 頁

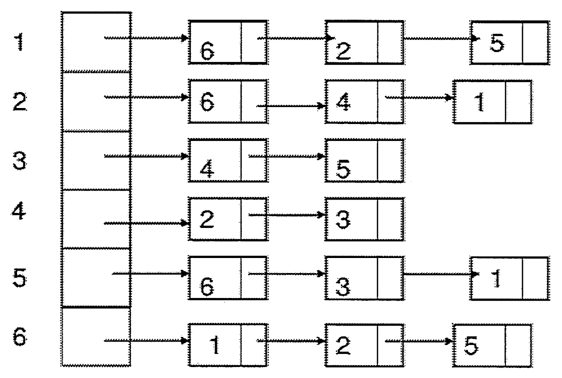
註：本次考試 <u>不可以</u>參考自己的書籍及筆記； <u>不可以</u>使用字典； <u>不可以</u>使用計算器。

1. (10 points)

The figure on the right is a graph represented by an adjacency list.

(i) Do *BFS* (*breadth first search*) on the graph and show the visited vertices in order (i.e., the 1st visited vertex number, the 2nd visited vertex number, the 3rd visited vertex number, and so on) if the starting vertex is 3.

(ii) Do *DFS* (*depth first search*) on the graph and show the visited vertices in order if the starting vertex is 3.



2. (20 points)

Draw the 11-item *hash table* resulting from hashing the keys **<u>1, 11, 13, 22, 12, 6, 33, 39, 20, 5, 16</u>** by using the hash function **<u>$h(k) = (2k + 5)$ mod 11</u>**. The collisions are handled by (i) *linear probing* and (ii) *double hashing* with the secondary hash function: **<u>$h'(k) = 7 - (k \bmod 7)$</u>**.

3. (10 points)

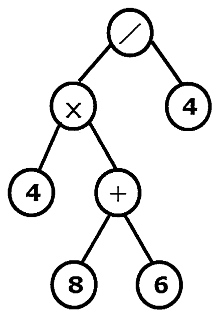
State (i) the difference between *stacks* and *queues*, and (ii) the difference between *queues* and *priority queues*.

4. (5 points)

What are the advantages of *height-balanced binary trees* over normal *binary search trees*?

5. (15 points)

There are three common tree traversals: *preorder* traversal, *inorder* traversal, and *postorder* traversal. Show the three traversal results of the *binary tree* shown on the right.



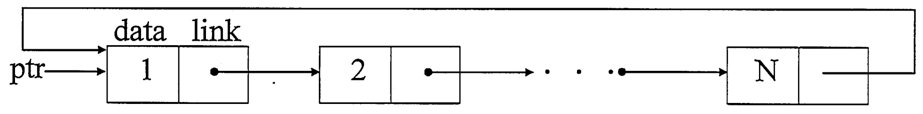
6. (10 points)

Write the worst case running time (in **<u>big-O notation</u>**) for each of the following operations: (i) searching an *unordered list* with *n* items; (ii) searching an *ordered list* with *n* items; (iii) searching a *binary search tree* with *n* nodes; (iv) searching a *height-balanced binary search tree* with *n* nodes; (v) searching a *hash table* with *n* keys.

7. (30 points)

The following C declarations define the node structure of a *linked list*.

```
typedef struct list_node *list_pointer;
struct list_node {
    int data;
    list_pointer link; };
```



The figure on the upper right is an example of *circularly linked lists*, where `ptr` points to the first *node* of the *list*. According to the above C declaration, write an **<u>iterative</u>** C function, `interPrintNode( list_pointer ptr )`, and a **<u>recursive</u>** C function, `recurPrintNode( list_pointer ptr )`, that can print the `data` of the *nodes* sequentially in a *circularly linked list*. Moreover, write a C function, `deleteFirstNode( list_pointer* ptr )`, that can delete the first *node* of the *circularly linked list*.

<<以下空白>>