

This question paper must be returned. Candidates are not permitted to remove any part of it from the examination room.

STUDENT'S FAMILY NAME

OTHER NAMES

STUDENT NUMBER

Date of Birth

Address

.....
(MUST be completed for identification)

MACQUARIE UNIVERSITY

Unit : COMP225 - Algorithms and data structures

Date : Midterm Test, May 2009

Time Allowed : 45 minutes

Instructions : Answer ALL questions.
Answers must be written on this paper in the space provided;
: no additional sheets will be accepted.

1. **(10 marks total)** Complete the C++ function definitions for the specifications given below. For each function you write state its worst case time complexity assuming that the tree has n nodes. You may also assume that t is a pointer to a binary tree (NOT a binary *search* tree unless otherwise stated). The binary tree type uses the following definition.

```
struct treeNode {
    int item;
    treeNode* left;
    treeNode* right;
};
```

- (a) **(2 marks)**

```
int height(treeNode* t)
// pre : t is a pointer to a binary tree
// post : returns the height of t
```

Worst Case complexity is:

- (b) **(2 marks)**

If t contains items 1, 3, 6 then `product` returns $1 \times 3 \times 6 = 18$.

```
int product(treeNode* t)
// pre : t is a pointer to a binary tree
// post : returns the product of the items in t
```

Worst Case complexity is:

(c) (2 marks)

If t contains items 1,5,3,7,8,10, then $\text{maxVal}(t, 4) = 5$, and $\text{maxVal}(t, 7) = 7$.

```
int maxVal(treeNode* t, int k)
// pre : t is a pointer to a binary tree, k is an integer;
//       all items in t are non-negative
// post : returns the smallest item in t which is at least the value of k;
//       returns -1 if there is no such item in t.
```

Worst Case complexity is:

(d) (3 marks) A tree is *full* if it is (i) empty OR (ii) both the left and the right subtrees are full AND they have the same number of nodes as each other.

Write a recursive function to calculate whether a tree is full or not. (HINT: You might need to specify an auxiliary function to compute the number of nodes. If you do, then give its specification, but do not implement it.)

```
bool full(treeNode* t)
// pre : t is a pointer to a binary tree
// post : returns true if t is full, and false otherwise.
```

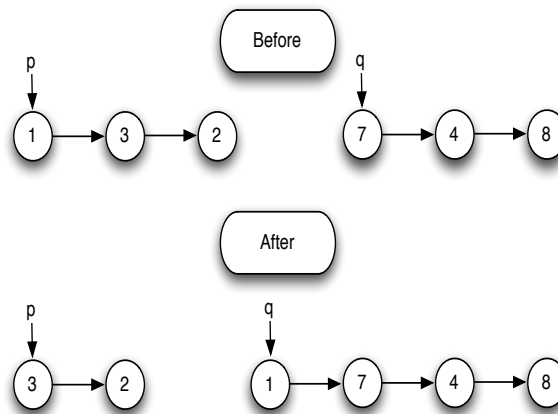
Worst Case complexity is:

(e) (1 mark) Which of the above functions (if any) could be made more efficient if t is a binary search tree?

2. (10 marks total) Consider the following definition for a node in a linked list.

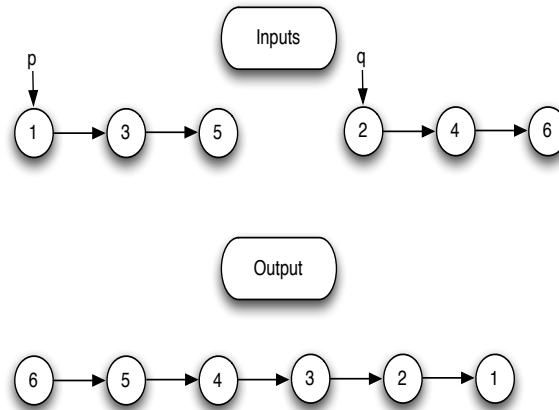
```
struct node {  
    int item;  
    node* next;  
};
```

(a) (5 marks) Write a function which takes a node from the front of one list and puts it on the front of the other. For example



```
void takePut(node* &p, node* &q)  
// pre : p, q are pointers to linked lists  
// post : takes the top node from p and puts it to the front of q
```

- (b) (5 marks) Write a function which takes two sorted lists and merges them in the opposite order. Your solution should use $O(1)$ scratch space. (You must use `takePut` in your implementation to reassign pointers if necessary. Do not use any other method for reassigning pointer values.) For example:



```
node* antiMerge(node* &p, node* &q)
// pre : p, q are pointers to sorted linked lists
// post : returns the merge of p and q in reverse order; Uses  $O(1)$  space
```