1. (10%) Consider a *linked list*. A node in the list stores a positive integer and has a *pointer* named 'next' pointing to the next node. Write a **recursive** function that returns the largest value of a given linked list and has all the nodes with that value removed from the list after completing the function call. Your function should take a pointer, which points to a node of an existing linked list, as one of the arguments.
2. (10%) Consider a *linked list*. A node in the list has a *pointer* named 'next' pointing to the next node. Write a **non-recursive** function that frees the nodes in even positions of a given linked list (the second, fourth, sixth, and so forth). Your function should take a pointer, which points to the head node of an existing linked list, as the argument.
3. (10%) Consider a *circular linked list*. A node in the list has a pointer named 'next' pointing to the next node. Write a function that returns the number of nodes on a circular list. Your function should take a pointer, which points to a node of an existing circular linked list, as one of the arguments.
4. (10%) Consider a *binary tree*. A node in the binary tree stores an integer and has two pointers named 'left' and 'right'. Write a **recursive** function that implements a **postorder** traversal. A **postorder** traversal visits the left subtree first, then visits the right subtree, and finally visits the current node. Visiting a node means printing the integer stored in the node.
5. (10%) Consider a *binary tree*. A node in the binary tree stores an integer and has two pointers named 'left' and 'right'. Using a *stack* that pushes and pops *pointers*, write a **non-recursive** function that takes a pointer pointing to a tree node as an argument and traverses the tree in **preorder** way. A **preorder** traversal visits the current node first, then visits the left subtree, and finally visits the right subtree. Visiting a node means printing the integer stored in the node.
6. (10%) Consider an *undirected graph*. Prove that, if a connected graph of N nodes has the property that removing any edge disconnects the graph, then the graph has $N - 1$ edges and no cycles.
7. (10%) Consider a *directed graph*. Suppose that you have a set of nodes with no null pointers (each node points to itself or to some other node in the set). Prove that you ultimately get into a cycle if you start at any given node and follow links.
8. (10%) What is a *heap*? How to construct a heap in linear time.
9. (10%) How can a *heap* improve the performance of a *priority queue* when compared with using an *array* or *list* implementation?
10. (10%) Using a *stack* that pushes and pops *characters*, write a function that reads in a sequence of characters stored in a 'character' *array*, and determines whether its parentheses, braces, and curly braces are balanced. For example '([])' is balanced but '([)]' is not.

試題隨卷繳回