CITY UNIVERSITY OF HONG KONG

Course code & title	:	EE2331 Data Structures and Algorithms
Session	:	Semester B 2011/12
Time allowed	:	Two hours

This paper has SEVEN pages (including this cover page).

---

1. This paper consists of 9 questions.
2. Answer ALL questions.

---

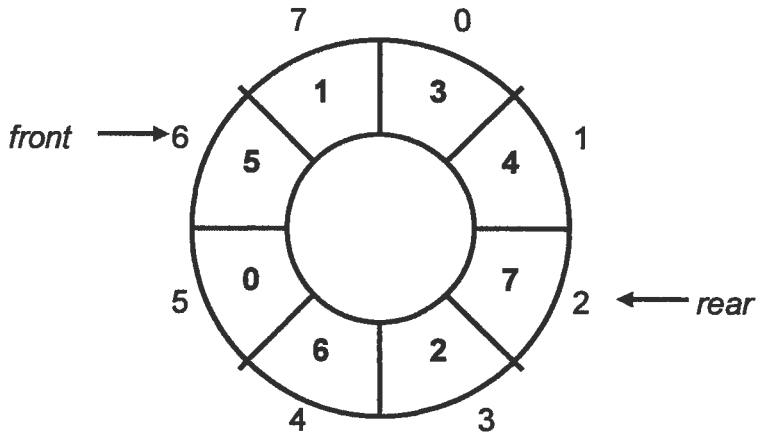
*This is a **closed-book** examination.*

Candidates are allowed to use the following materials/aids:

Approved Calculators

Materials/aids other than those stated above are not permitted. Candidates will be subject to disciplinary action if any unauthorized materials or aids are found on them.

Question 1: Given the following queue q represented by a circular array with *front* = 6 and *rear* = 2:



(a) Draw the array, with *front* and *rear* and show the output after a dequeue operation is done to q. **(3 Marks)**

(b) Draw the array, with *front* and *rear* after an enqueue(9) operation is done to the original q. **(3 Marks)**

(c) How to determine if a queue represented by a circular array is full? **(2 Marks)**

(d) How to determine if a queue represented by a circular array is empty? **(2 Marks)**

(e) How many elements can be stored in the above queue? Why? **(2 Marks)**

Question 2: Trace the execution of a recursive binary search for *key* = 18 in the following array

{2, 3, 5, 7, 11, 13, 17, 19, 23, 29}

Complete the following table:

Recursive Call	*lo*	*mid*	*hi*
1			
2			

⋮

(4 Marks)

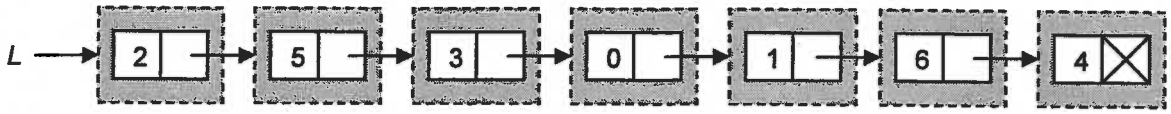
Question 3: The structure of a linked list node is defined as follows:

```
struct _node {
    int data;
    struct _node *next;
}
typedef struct _node Node;
```

Consider a recursive function **recur** for linked list:

```
Node * recur(Node *p) {
    Node *q;
    if (p != NULL && p->next != NULL) {
        q = p;
        p = p->next;
        q->next = p->next;
        p->next = q;
        q->next = recur(q->next);
    }
    return p;
}
```

The initial contents of a singly linear linked list *L* are shown below:



Draw the list *L* after running the following statement:

```
L = recur(L);
```

(4 Marks)

Question 4: Consider a recursive function **recur**:

```
void recur(int x) {
    if (x <= 1) return;
    if (x % 2) recur(3 * x + 1);
    printf("%d  ", x);
    recur(x / 2);
}
```

Trace the outputs for:

```
recur(3);
```

(6 Marks)

Question 5: Given the following array A:

{90, 61, 10, 52, 23, 34, 24, 24*}

(a) The merge sort algorithm is applied to sort the array A in ascending order. Show the array contents after each pass.

(4 Marks)

(b) The standard quicksort algorithm is applied to sort the array A in ascending order. Show the array contents after each pass.

(6 Marks)

(c) Merge sort and quicksort are both O($n$log$n$) algorithms. Give one situation that you would prefer to sort the data using:

(i) merge sort
(ii) quicksort

(4 Marks)

Question 6: Consider a hash table of size 11, and the keys are integers. The hash function is $h(key) = key \% 11$. Collisions are resolved by quadratic probing. The initial contents of the hash table are shown below.

0	
1	23
2	90
3	
4	
5	34
6	61
7	
8	
9	16
10	10

(a) What is the average number of probes for successful search? **(4 marks)**

(b) Show the contents of the hash table after the ***key* = 100** has been inserted. **(3 marks)**

(c) What is the problem of using quadratic probing to handle collision? Please use one to two sentences to briefly describe it. **(3 marks)**

Question 7: Given the stack definition and operations as follows:

```
typedef struct _stack Stack;          //the stack structure
void stack_init(Stack *s);            //initialize s to empty
void stack_destroy(Stack *s);         //delete all elements from s
void push(Stack *s, int e);           //insert an element e to the top of s
int pop(Stack *s);                    //remove and return the top element
int stak_is_empty(Stack *s);          //return 1 if s is empty, or 0 if non-empty
```

An integer stack A is initialized to contain n numbers where $n \geq 0$.

$e_1, e_2, e_3, \ldots e_n$
(where e_1 is the bottom element, e_n is the top element)

We would like to rearrange the elements in A to the following order:

$e_1, e_3, e_5, \ldots e_n \ldots e_6, e_4, e_2$
(where e_1 is the bottom element, e_2 is the top element)

Write a non-recursive function to rearrange the elements in A, with the help of two additional stacks.

```
void rearrange(Stack *A);
//You may use at most one integer variable and two extra stacks only.
//Creating extra variables nor arrays is not allowed.
```

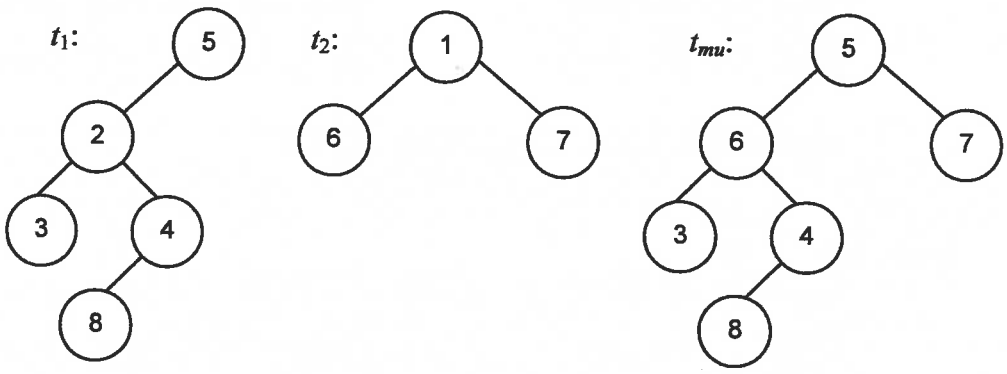
(10 Marks)

Question 8: The structure of the tree node of a binary tree is defined as follows:

```
struct _treenode {
   int data;
   struct _treenode *left, *right;
}
typedef struct _treenode TreeNode;
```

(a) Given 2 binary trees t_1 and t_2, the *Max-Union* of t_1 and t_2 is a binary tree t_{mu} such that:
(1) if t_1 and t_2 are both NULL, t_{mu} is NULL
(2) if t_1 is NULL and t_2 is non-NULL, t_{mu} is the same as t_2.
(3) if t_2 is NULL and t_1 is non-NULL, t_{mu} is the same as t_1.
(4) if t_1 and t_2 are both non-NULL, the root of t_{mu} contains the value that is the maximum between the root nodes of t_1 and t_2, and the left subtree of t_{mu} is the *Max-Union* of the left subtrees of t_1 and t_2, and the right subtree of t_{mu} is the *Max-Union* of the right subtree of t_1 and t_2.

For example:



Write a recursive function to create the *Max-Union* of two binary trees. You may assume **stdlib.h** has been included and you should use the function **malloc** to allocate memory when needed. **(10 Marks)**

```
TreeNode * max_union(TreeNode *t1, TreeNode *t2);
```

(b) A general tree can be represented using a binary tree. Write a recursive function to compute and return the depth of a general tree.

(10 Marks)

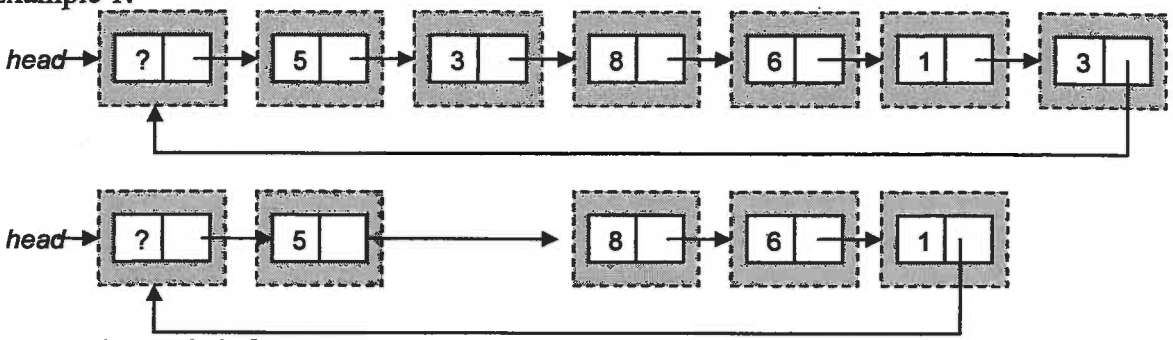
```
int depth(TreeNode *p);
```

Question 9: In this question, a singly circular linked list with dummy header is used to represent a list of integers. The structure of a linked list node is defined as follows:

```
struct _ node {
    int data;
    struct _node *next;
}
typedef struct _node Node;
```

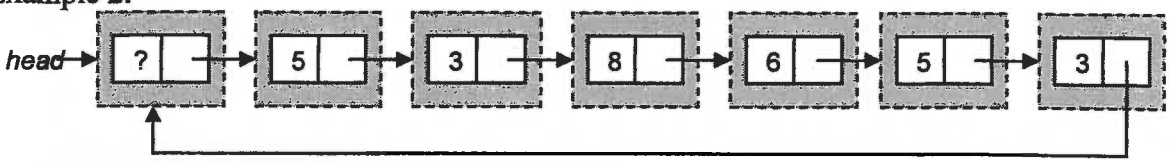
The *mode* of a list of n numbers ($n \geq 0$) is the number m in the list that is repeated most frequently. If more than one number is repeated with equal maximal frequencies, there is no *mode*.

Example 1:



The mode is 3. Two nodes have been deleted.

Example 2:



There is no mode (both 5 and 3 appear twice). No nodes to be deleted.

Write a non-recursive C function to find and delete the *mode* in the list. The function should return 1 to indicate the *mode* has been found and deleted. If there is no *mode*, return 0.

You may assume **stdlib.h** has been included and you should use the function **free** to deallocate memory when needed.

```
int find_and_delete_mode(Node *head, int *m);
```

(20 Marks)

- THE END -