City University of Hong Kong

---

Course Code & Title: EE2331 Data Structures and Algorithms

Semester/year: Semester B, 2008-09

Time Allowed: Two hours

---

This paper has 6 pages (including this cover page)

Answer ALL questions in this paper.

---

Material, aids & instruments permitted to be used during examination:

1. Approved calculators

**Question 1 (20 marks)**

The node structure of a singly linked list is defined below.

```
typedef struct _node {
    int SID;                  //the ID of the student
    int cohort;               //the cohort year
    char name[80];            //the student name
    float CGPA;               //the GPA of the student
    int marks[MAX_COURSE];    //the marks of his courses
    struct _node * next;      //point to next student record
} Node;
```

(a) Implement the C function `extract()` that accepts a pointer to the head of a linked list (*head*), two floating point numbers (*a*, *b*) and extracts the nodes in the input list where the CGPA value is greater than or equal to *a* and is less than or equal to *b*. The extracted nodes will be removed from the original list and will form a new list with dummy header and the reference to the new list will be returned to the calling function. The sequence order of the nodes should be retained.

You should manipulate the pointers only, and should not copy the values from one node to another. You may create local variables, and use the malloc() function to allocate memory if necessary. **(16 marks)**

```
Node* extract(Node *head, float a, float b) {
    //Precondition:
    //The input list is unordered and has dummy header node
}
```

(b) How would you modify your function in (a) if the input linked list (not output linked list) does not have dummy header node? You do not have to write any code. Please state the part(s) that need to be changed, how to change and explain why. **(4 marks)**

**Question 2 (30 marks)**

A recursive pattern $P(n)$ is defined as follows:

$P(n) = (P(n\text{-}1), P(n\text{-}1), n)$ for $n > 1$

$P(1) = (1)$ for $n = 1$

For example:

$P(1)$ will output "1 "

$P(2)$ will output "1 1 2 "

$P(3)$ will output "1 1 2 1 1 2 3 "

... etc

S($n$) is defined as the sum of the sequence $P$(n).

$S(1) = 1$

$S(2) = 4$

$S(3) = 11$

...etc

(a) Design a recursive C function **void printP(int n)** that accepts an integer $n$ and prints the pattern of $P(n)$ to screen. The numbers are separated by space, and the double quotation "" is not required. The pattern is undefined for $n < 1$. So please print nothing for the invalid $n$ values. *Note: there is always a space at the end of the pattern!*

**(5 marks)**

```
void printP(int n) {


}
```

(b) Design an **O($n$)** recursive C function **int S(int n)** that returns the value of $S(n)$. You should minimize the number of recursive calls in your function. **(5 marks)**

```
int S(int n) {


}
```

*(Continued on next page)*

(c) Given the prototype of queue as follows. The actual implementation of the queue is unknown to you.

```
typedef struct _queue Queue;      //the name of the queue structure
void Queue_init(Queue *q);        //to init the queue to empty state
void Queue_destroy(Queue *q);     //to delete the data of queue
int Queue_isEmpty(Queue *q);      //return 1 if the queue is empty,
                                  //otherwise return 0
void enqueue(Queue *q, int e);    //to insert an element to the rear
int dequeue(Queue *q);        //to delete and return the front element
```

Implement the function **void copy(Queue *target, Queue *source)** that creates a copy of a queue. The target queue should have the same content as the source queue and the source queue should remain unchanged after running your function.

In your function, you should call the above provided queue functions if you want to get or delete any elements, or check the state of the queue. You should not try to go into the internal structure of the queue, nor create any array to store the elements of the queues. **(10 marks)**

```
void copy(Queue *target, Queue *source) {


}
```

(d) Design a non-recursive C function to print the $P(n)$ sequence. You may call any functions in part (c), if necessary. **(10 marks)**

Hints: you have to use two queues to help you.

$P(n)$ can be obtained with the following procedures:

Enqueue $P(n\text{-}1)$ sequence into *Queue 1*
Copy *Queue 1* to *Queue 2*
Dequeue $P(n\text{-}1)$ sequence from *Queue 2*, and enqueue into *Queue 1*
Enqueue $n$ into *Queue 1*, and
$P(n)$ is now in *Queue 1*

Starting from $P(1)$, you can then obtain $P(2)$, $P(3)$, up to $P(n)$.

```
void printP_nonrecur(int n) {


}
```

**Question 3 (20 marks)**

The node structure of a tree is defined below.

```
typedef struct _treenode {
    int data;                          //the value of the tree node
    struct _treenode *left, *right; //point to left & right child
} TreeNode;
typedef struct _tree {
    struct _treenode *root;            //the root node of the tree
} Tree
```

(a) Write a recursive function to test if the tree pointed by *p* is a BST. **(10 marks)**

```
int isBST(TreeNode* p) {
    //return 1 if the tree pointed by p is BST, otherwise return 0


}
```

(b) Write a recursive function to print the BST nodes whose degree equals to *k*. The node values should be output in descending order. **(10 marks)**

```
void print(TreeNode *p, int k) {


}
```

**Question 4 (16 marks)**

(a) Work through the steps of sorting the keys *7, 4, 1, 8, 5, 2, 9, 6, 3, 4* to ascending order using heapsort. Focus on the major steps and show the content of the array after each pass. You are not required to draw any diagram. **(12 marks)**

(b) Please indicate the stability of the sorting algorithm in (a) and explain your answer. **(4 marks)**

**Question 5 (14 marks)**

(a) Assuming that a hash table is of size 7, construct the hash table with the below data sequence while the hash function is defined as $h(i) = i \% 7$ and linear probing is applied,

23, 10, 3, 8, 15, 9

Show the contents of the hash table after the insertion of each element. **(8 marks)**

(b) Determine the average number of comparisons for a successful search. **(3 marks)**

(c) Suggest a modification of the design of the hash table in part (a) that would reduce the value obtained in part (b). Explain briefly. **(3 marks)**

**- THE END -**