國立臺灣科技大學
九十三學年度碩士班考試試題

系所組別：資訊工程系
科　　目：資料結構

總分 100 分

1.

(6%) 1.1 Let N_h represent the minimum number of elements in an AVL tree of height h. Why is the following formula true: $N_h = N_{h-1} + N_{h-2} + 1$ for $h \geq 2$.

(4%) 1.2 What is the maximum possible height of an AVL tree that contains 42 data elements?

1.3 An *external* node in a 2-3 tree refers to a nonexistent node that corresponds to a null link, while all the nodes that actually contain data elements are called *internal* nodes. Suppose that a 2-3 tree ***T1*** has a total of 8 internal nodes and 12 external nodes.

(3%) *(a)* How many data elements are there in the tree ***T1***?

(4%) *(b)* What is the maximum possible height of ***T1***?

2.

(4%) 2.1 What kind of sequence does *inorder* traversal of a binary search tree produce?

(5%) 2.2 Can we traverse a binary tree having *n* nodes with a time complexity better than *O(n)*? Why?

3. A max heap can be represented as an array. Let array `A[]` (shown below) be a max heap (`A[0]` is not used). Initially, `A[]` contains two elements: 8, 32. In the figure the elements of no concern are left blank.

(5%) 3.1 Show the resulting array `A[]` after the following three elements, 17, 44 and 68, are sequentially inserted into the max heap.

(5%) 3.2 Based on the result of problem 3.1, show the array `A[]` after the element 68 is deleted from the max heap.

32	8					
A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]

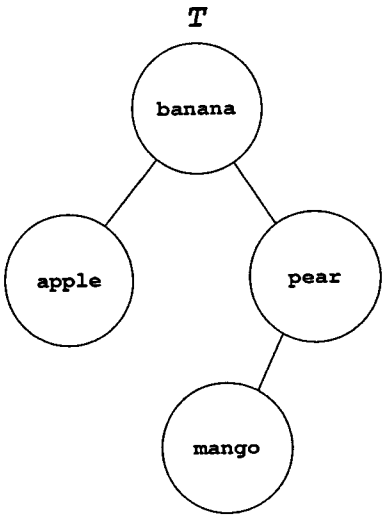
國立臺灣科技大學
九十三學年度碩士班考試試題

系所組別：資訊工程系
科　　目：資料結構

4. You are given a set of four data elements with keys: **apple**, **mango**, **banana**, **pear**. These data elements are organized as a binary search tree ***T*** as shown in the following figure. Suppose that the probabilities of searching for the data elements with keys **apple**, **mango**, **banana** and **pear** are 0.3, 0.08, 0.17 and 0.45, respectively. (Unsuccessful searches are not considered.)

(5%) 4.1 What is the average number of key comparisons with ***T***?

(9%) 4.2 If instead of a binary search tree we arrange the same four data elements as a singly linked list. We then use sequential search to search the data elements. Assume that the search probabilities for the keys remain the same. Does there exist a singly linked list that offers an average number of key comparisons that is smaller than the one offered by ***T***? If yes, give one such list.



國立臺灣科技大學
九十三學年度碩士班考試試題

系所組別：資訊工程系
科　　目：資料結構

5. (10%) We would like to do mergesort by a bottom up manner. When the input has 2^N elements, we can use the following program to complete the sorting:

```
MERGESORT(A, p, r)
1  L ← r - p + 1
2  N ← 1
3  while N ≤ L/2
4     for i ← 0 to L/(N*2)-1
5        MERGE(A, i*N*2+1, (i*2+1)*N, (i+1)*N*2)
6     N ← N + N

MERGE(A, p, q, r)
1   n1 ← q - p + 1
2   n2 ← r - q
3   create arrays L[1 .. n1 + 1] and R[1 .. n2 + 1]
4   for i ← 1 to n1
5      do L[ i ] ←A[p + i - 1]
6   for j ← 1 to n2
7      do R[ j ] ←A[q + j]
8   L[n1 + 1] ← ∞
9   R[n2 + 1] ← ∞
10  i ←1
11  j ←1
12  for k ← p to r
13     do if  L[ i ] ≤ R[ j ]
14          then A[ k ] ← L[ i ]
15               i ← i + 1
16          else A[ k]  ← R[ j]
17               j  ← j + 1
```

If the input does not have the number of 2^N, can you come up with some simple solution to do the bottom-up mergesort. (No need to write a program, just describe your idea, possible by giving an example).

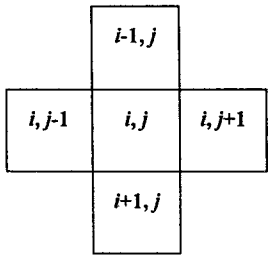
國立臺灣科技大學
九十三學年度碩士班考試試題

系所組別：資訊工程系
科　　目：資料結構

6. (10%) We have a recursive program called *paint* which can paint a region with the same color:

```
void paint(int i, int j, color)
{
    if (inside(i-1, j) && !visited(i-1, j)) paint(i-1, j, color);
    if (inside(i, j+1) && !visited(i, j+1)) paint(i, j+1, color);
    if (inside(i+1, j) && !visited(i+1, j)) paint(i+1, j, color);
    if (inside(i, j-1) && !visited(i, j-1)) paint(i, j-1, color);
}
```

where the function *inside*(int *i*, int *j*) can judge if the location is still inside the region or not and the function *visited*(int *i*, int *j*) will output "yes" if the cell has been visited before. The coordinate system is given by (a) in the following figure:



(a)

7	6	5	4	13
8	9	1	3	14
12	10	11	2	15

(b)

Given the 3x5 rectangle in (b), starting from the black cell, please write down numbers in all cells to describe order of the visiting by the *paint* program. For instance, (b) shows a possible visiting (clearly not the answer).

7. Let us consider several problems related to polynomial.
(5%) (a) If we want to evaluate a polynomial $f(x) = 3x^5-5x^4+2x^3+9x^2-10x+1$, given x equal to an integer, in what way, we can apply as least multiplication as possible in the calculation. Please write down the answer by putting parentheses in the formula (if necessary). For instance, we can put (6+2)*3 to indicate a summation of 6+2 is applied first and a multiplication of 8*3 is applied afterwards. How many additions/subtractions and/or multiplications do we use in the calculation?
(5%) (b) Same question is asked again, but for the polynomial $g(x) = 3x^9-x^8+1$. How to evaluate $g(a)$, using as few multiplications as possible? How many additions/subtractions and multiplications do we need?
(5%) (c) Please give the link-list representation of $g(x)$, using the space economically.

8. Considering two data structures to represent a directed graph $G = (V, E)$: (1) adjacency matrix and (2) adjacency lists, please provide the most efficient (fast) structure for the following situations.
(5%) (a) We want to find if there is any vertex v or not such that every other vertices in the graph point to this vertex v, i.e., we have edge $e = u \rightarrow v$, $\forall u \in V$. Which data structure is more efficient than the other, (1) adjacency matrix or (2) adjacency lists? Please give a brief explanation.
(5%) (b) We want to find if there is any vertex u or not such that no edge in the graph starts from u, i.e., no edge with the form $e = u \rightarrow v$. Which data structure is more efficient than the other, (1) adjacency matrix or (2) adjacency lists? Please give a brief explanation.

國立臺灣科技大學
九十三學年度碩士班考試試題

系所組別：資訊工程系
科　　目：資料結構

9. (5%) We can use Kruskal algorithm to find the minimum spanning tree. Is it possible that Kruskal algorithm finds a spanning tree with summation of all tree edges, e.g, equal to A+2+4 and another minimum spanning tree (possibly computed by a different method) gives you the summation of all tree edges as A+3+3 while both trees are minimum spanning trees? (A is a summation of possibly more than one term, e.g., A=10+5+7.) Please write down only the answer without any proof.