類組：電機類　科目：資料結構(3002)　　共 8 頁 第 1 頁

第一大題 單選題，每題3分，答錯不倒扣，共30分。

Multiple choice (single answer) questions (3 points per question)

1. Let A be an array of n distinct numbers. If i<j and A[i]>A[j], then the pair (i, j) is called an inversion of A. What is the expected number of inversions in any permutation on n elements?

(A) n(n-1)/2

(B) n(n-1)/4

(C) n(n+1)/4

(D) 2n(logn)

(E) n

2. Five people A, B, C, D, and E are standing in a queue. A is standing behind C. B is not standing next to D. D and E are standing next to each other. Who is not possibly standing third in the queue?

(A) A

(B) B

(C) C

(D) D

(E) E

3. Which sorting algorithm listed below possesses the highest worst-case time complexity?

(A) Merge Sort

(B) Quick Sort

(C) Heap Sort

(D) Radix Sort

4. Which sorting algorithm typically requires the most amount of additional memory space (i.e., has the highest space complexity)?

(A) Insertion Sort

(B) Selection Sort

(C) Merge Sort

(D) Heap Sort

注意：背面有試題

5. What should be added to the blank so that the following function can correctly reverse a singly linked list.

```
struct node {
    int data;
    struct node * next;
};
static void reverse(struct node ** head) {
    struct node * prev = NULL;
    struct node * current = *head;
    struct node * next;
    while (current != NULL) {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    ____________________________
}
```

(A) *head = prev;
(B) *head = current;
(C) *head = next;
(D) *head = NULL;
(E) None of the above

6. After partitioning in a Quick Sort, the array (20, 7, 12, 15, 6, 11, 9) becomes (7, 6, 9, 20, 12, 15, 11). Which following statement is correct?
(A) The pivot could be 6.
(B) The pivot could be 9.
(C) The pivot could be 12.
(D) The pivot could be 20.

7. A sorting algorithm is considered stable if:
(A) It requires O(n^2) extra space for sorting.
(B) It employs a divide and conquer strategy for sorting.
(C) It maintains the relative order of equal elements as they appeared in the original array.
(D) It sorts the elements in O(log n) time complexity.

注意：背面有試題

8. Consider the following Python codes, which traversal order will the codes produce when starting from node 'A' in the given graph?

```
def search(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)
    print(start, end=' ')
    for neighbor in graph[start]:
        if neighbor not in visited:
            search(graph, neighbor, visited)
graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': []
}
```

(A) A B D E F C
(B) A C F B D E
(C) A B E F C D
(D) A B D E C F
(E) A D F C B E

9. Which search concept is the code in question 10 implemented for?
(A) Breadth-first search
(B) Depth-first search
(C) Binary search
(D) Linear search
(E) None of the above

10. What is the minimum number of edges in an undirected tree with 10 vertices that contains a vertex with a degree of 4?
(A) 7
(B) 8
(C) 9
(D) 10
(E) 11

注意：背面有試題

第二大題 多選題

multiple choice questions with one or more answers (5 points per question) 每題每一選項(ABCDE)單獨計分，每一選項個別分數為 1 分，答錯倒扣 1 分，倒扣至本大題（即多選題）0 分為止。

11. If there are two nodes such that their “next” pointers point to the same node in a singly linked list structure, what configurations are possible?
 (A) There are two normal singly linked lists
 (B) There is a loop in the linked list
 (C) It is impossible that two nodes can point to the same node as their next node
 (D) This is a cyclical list structure
 (E) Two singly linked lists intersect

12. Below is an algorithm in pseudocode that sorts a stack of integers. S1 is the input stack and S2 is an additional temporary stack for storing and moving items. The stack supports the following operations: push, pop, peek, and isEmpty.

```
void sort(S1){
    S2 = new Stack<integer>;//an empty stack for integers
    while(!S1.isEmpty()){
        int tmp = S1.pop();
        while(!S2.isEmpty() && S2.peek() > tmp) {
            S1.push(S2.pop());
        }
        S2.push(tmp);
    }
    While (!S2.isEmpty()) {
        S1.push(S2.pop());
    }
}
```

Which of the following is correct?
(A) The algorithm is O(nlogn) time-efficiency
(B) The algorithm is O(n) space-efficiency
(C) The algorithm sorts the stack in ascending order
(D) The algorithm sorts the stack in descending order
(E) The algorithm is O(n^2) time-efficiency

13. Which of the following statements are true about Merge Sort? (Select all that apply.)
   (A) It has a worst-case time complexity of O(n log n).
   (B) It is not a stable sorting algorithm.
   (C) It is more efficient than Quick Sort in the worst-case scenario.
   (D) It requires additional space proportional to the size of the input array.
   (E) It uses O(n log n) extra space for its operations.

14. Consider the following statements about hashing. Which of these are correct? (Select all that apply.)
   (A) Chaining as a collision resolution technique can degrade to O(n) in the worst case.
   (B) Hash functions should ideally be dependent on the size of the input data.
   (C) Reducing the size of a hash table will generally improve its performance.
   (D) A hash table with a larger load factor is always more efficient.
   (E) Hash functions should ideally distribute the data uniformly across the hash table.

15. In the context of graph theory and finding optimal paths or trees, which statements accurately describe features of Dijkstra's, Prim's, and Kruskal's algorithms? (Select all that apply)
   (A) Dijkstra's Algorithm operates on graphs with non-negative edge weights only.
   (B) Kruskal's Algorithm guarantees finding a minimum spanning tree for both directed and undirected graphs.
   (C) Prim's Algorithm selects the next node based on the lowest edge weight.
   (D) Dijkstra's Algorithm guarantees finding the longest path from a single source to all other vertices in a graph.
   (E) Prim's Algorithm guarantees the creation of a minimum spanning tree starting from an arbitrary vertex.

16. In the context of Red-Black Trees, which of the following statements are false? (Select all that apply)
   (A) Red-Black Trees ensure a maximum height difference of 2 between any two leaf nodes.
   (B) They are a type of self-balancing binary search tree.
   (C) Red-Black Trees can degenerate to a linked list during certain insertions.
   (D) The root of a Red-Black Tree is always red.
   (E) Red-Black Trees require additional memory compared to regular binary search trees due to color information stored with each node.

第三大題 問答題

Short description question (10 points per question)

1. (A) The minimum number of comparisons required to determine if an integer appears more than n/2 times in a sorted array of n integers is _________ (O(n), O(logn), O(n*logn), or O(1)). (2 points)

   (B) Write a solution in C language as a function to find how many times an integer "key" appears in a sorted array A sized N. (8 points) (Code is graded based on correctness and computational efficiency)(A is sorted from lowest to highest).

```
int Func(const int *A, int N, int key);
//input: an integer array A sized N, key is the integer to be searched
//output: integer value "key" appearance times
```

2. Analyze the characteristics and functionalities of hash tables, where α is the load factor of the hash table.

   (A) Which collision resolution technique can lead to primary clustering? (3 points)

   (B) What is the average-case time complexity of search operations in a hash table using chaining for collision resolution, in Big-O notation, assuming a good hash function? (3 points)

   (C) Consider a hash table using linear probing for collision resolution where n is the number of elements in the hash table. What happens to the average-case time complexity of search operations as the load factor α approaches 1? (4 points)

3. Given the following codes, please answer the three questions

   (A) What is the algorithm that the codes are implemented for? (3 points)

   (B) What does the graph look like? (3 points)

   (C) What is the output of the codes? (4 points)

```
import java.util.*;
class process {
    private static final int V = 5;
    int minKey(int key[], Boolean mstSet[]) {
        int min = Integer.MAX_VALUE, min_index = -1;
        for (int v = 0; v < V; v++)
            if (!mstSet[v] && key[v] < min) {
```

注意：背面有試題

```
                min = key[v];
                min_index = v;
            }
        return min_index;
    }
    void print(int parent[], int graph[][]) {
        System.out.println("Edge \tWeight");
        for (int i = 1; i < V; i++)
            System.out.println(parent[i] + " - " + i + "\t" + graph[i][parent[i]]);
    }
    void dowork(int graph[][]) {
        int parent[] = new int[V];
        int key[] = new int[V];
        Boolean mstSet[] = new Boolean[V];
        for (int i = 0; i < V; i++) {
            key[i] = Integer.MAX_VALUE;
            mstSet[i] = false;
        }
        key[0] = 0;
        parent[0] = -1;
        for (int count = 0; count < V - 1; count++) {
            int u = minKey(key, mstSet);
            mstSet[u] = true;
            for (int v = 0; v < V; v++)
                if (graph[u][v] != 0 && !mstSet[v] && graph[u][v] < key[v]){
                    parent[v] = u;
                    key[v] = graph[u][v];
                }
        }
        print(parent, graph);
    }
    public static void main(String[] args) {
        process p = new process();
        int graph[][] = new int[][] {
                    { 0, 2, 0, 6, 0 },
                    { 2, 0, 3, 8, 5 },
                    { 0, 3, 0, 0, 7 },
                    { 6, 8, 0, 0, 9 },
                    { 0, 5, 7, 9, 0 } };
        p.dowork(graph);
    }
}
```

4. Given the following codes, please answer the three questions:

(A) What should be filled in Q1 and Q2 (4 points)

(B) Which traverse order does the codes implement for? (3 points.)

(C) What is the output of the codes? (3 points.)

```
class Node {
    int key;
    Node left, right;
    public Node(int item) {
        key = item;
        left = right = null;
    }
}
public class BinarySearchTree {
    Node root;
    BinarySearchTree() {
        root = null;
    }
    void insert(int key) {
        root = insertRec(root, key);
    }
    Node insertRec(Node root, int key) {
        if (root == null) {
            root = new Node(key);
            return root;
        }
        if ( Q1 )
            root.left = insertRec(root.left, key);
        else if ( Q2 )
            root.right = insertRec(root.right, key);
        return root;
    }
    void order() {
        orderRec(root);
    }
    void orderRec(Node root) {
        if (root != null) {
            orderRec(root.left);
            System.out.print(root.key + " ");
            orderRec(root.right);
        }
    }
    public static void main(String[] args) {
        BinarySearchTree tree = new BinarySearchTree();
        tree.insert(5);
        tree.insert(3);
        tree.insert(2);
        tree.insert(4);
        tree.insert(7);
        tree.insert(6);
        tree.insert(8);
        tree.order();
    }
}
```