國立臺灣師範大學106學年度碩士班招生考試試題

科目：軟體基礎　　適用系所：資訊工程學系

注意：1.本試題共 7 頁，請依序在答案卷上作答，並標明題號，不必抄題。2.答案必須寫在指定作答區內，否則依規定扣分。

1. It is required to support the Search, Insert, and Delete operations after constructing a symbol table for a compiler. Please answer the following questions ((1)~(5)) according to the properties of the following 5 data structures ((a)~(e)) when they are used to construct a symbol table. Note that the answer of each question may contain more than one data structure. (10 分)

   Data structures:

   (a) An unordered linear list stored in an array.

   (b) An ordered linear list stored in an array.

   (c) An ordered linear list stored in a linked list.

   (d) A binary search tree. (Suppose that the binary search tree is highly balanced.)

   (e) A hash table. (Suppose that the size of the hash table is much adequately larger than the maximum number of possible entries. Therefore, very few data collision occurs.)

   Questions:

   (1) Which data structure can apply the binary search algorithm to efficiently search a data?

   (2) Which data structure the time complexity O(1) to perform the Insert operation?

   (3) Which data structure has the time complexity O(log n) to perform the Search operation when there are n data stored in the symbol table?

   (4) In general, which data structure has the worst performance when searching a data in the symbol table?

   (5) In general, which data structure is the most commonly used one for constructing a symbol table of a compiler?

2. (12 分)

   (1) The following 6 functions use C++ language to implement the Add or Delete operations for Stack, Queue, and circular Queue, respectively. However, the variable names didn't show their corresponding semantics. Please answer the following questions by choosing one of the given functions.

```
element function1(int *p, int r)
{
    if ( *p == r)
        return empty( );
    return arrayX[++ *p];
}
```

```
element function2(int *p)
{
    if (*top == -1)
        return empty( );
    return arrayX [(*p)--];
}
```

```
void function3(int *p, element item)
{
    if (*p >= MAX_SIZE-1)   {
                full( );
                return;
        }
        arrayX[++*p] = item;
}
```

```
void function4 (int p, int *r, element item)
{
       *r = (*r +1) % MAX_SIZE;
        if (p == *r)
        return;
    }
        arrayX [*r] = item;
}
```

```
element function5 (int* p, int r)
{
    element item;
    if (*p == r)
              return empty( );
   *p = (*p+1) % MAX _SIZE;
         return arrayX [*p];
}
```

```
void function6(int *r, element item)
{
      if (*r == MAX_ SIZE_1) {
           full( );
           return;
    }
    arrayX[++*r] = item;
}
```

(i) Which function is the most possible one to perform the Add operation for a stack?

(ii) Which function is the most possible one to perform the Delete operation for a queue?

(iii) Which function is the most possible one to perform the Add operation for a circular queue?

(2)

(i) Which data structure (stack or queue) is required to find a path of the mazing problem?

(ii) Which data structure (stack or queue) is required to translate an infix expression to a postfix expression?

(iii) Which data structure (stack or queue) is required to perform breadth first search on a graph.

3. (10 分)

(1) The following is a function in C++ language to implement the quicksort algorithm by recursion, where the data set for sorting is stored in the array **list** from entry index "left" to "right". Besides, the function SWAP(X, Y) will exchange the values of the two input variables X and Y. Please fill in the blanks (i), (ii), (iii), and (iv) to complete the function.

```
void quicksort(int list[], int left, int right)
{
   int pivot, i, j;
   if (left < right) {
      i = left;       j = right+1;
      pivot = list[left].key;
      do {
         do i++; while (list[i].key < pivot);
         do j--; while (list[j].key > pivot);
         if (i < j) SWAP (list[i], list[j]);
      } while (i < j);
      SWAP( ____(i)____, ____(ii)____ );
      quicksort(list, ____(iii)____, ____(iv)____ );
      quicksort(list, j+1, right);
   }
}
```

(2) Please explain the time complexity and give an example to describe the best case and worst case of performing the quick sort algorithm, respectively.

4. Given a cost matrix of an undirected graph as below, where each entry in **cost**[i][j] stores the cost of the edge (i, j) in the graph. The graph has 8 nodes and numbered from 0 to 7. If an edge does not exist, its corresponding entry in the matrix is ∞.

$$\begin{bmatrix} 0 & 5 & 2 & 3 & \infty & \infty & \infty & 7 \\ 5 & 0 & \infty & 5 & 1 & \infty & \infty & \infty \\ 2 & \infty & 0 & \infty & \infty & \infty & \infty & 3 \\ 3 & 5 & \infty & 0 & 3 & 6 & 4 & \infty \\ \infty & 1 & \infty & 3 & 0 & 2 & \infty & \infty \\ \infty & \infty & \infty & 6 & 2 & 0 & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty & \infty & 0 & \infty \\ 7 & \infty & 3 & \infty & \infty & \infty & \infty & 0 \end{bmatrix}$$

(1) Please find the bi-connected components of the graph. (4 分)

(2) Given the following function **shortestpath** in C++ language which can find the shortest distance from node *v* to the other nodes. Among the nodes whose corresponding values in the array **found** are labeled as **FALSE**, the function **choose_min**(distance, n, found) will return the index of the node with the minimum **distance** value.

According to the given cost matrix shown above, the function **shortestpath** is

called by setting v = 1. Please show the result of the array **distance** for each iteration of the **for** loop. (6 分)

```
void shortestpath(int v, int cost[][8], int distance[], int n, short int found[])
{
   int i, u, w;
   for (i=0; i<n; i++) {
      found[i] = FALSE;
      distance[i] = cost[v][i];
   }
   found[v] = TRUE;
   distance[v] = 0;
for (i=0; i<n-2; i++) {
      u = choose_min(distance, n, found);
      found[u] = TRUE;
      for (w=0; w<n; w++)
         if (!found[w])
            if (distance[u]+cost[u][w]<distance[w])
               distance[w] = distance[u]+cost[u][w];
   }
}
```

5. Given a sequence of numbers: 10, 3, 8, 12, 5, 7, 20, 4 to construct a minimum heap structure.

(1) The following function in C++ language inserts a number **item** into a minimum heap, where *n denotes the number of numbers which have been stored in the minimum heap. Initially, *n = 0.

Please show the constructed min_heap after inserting the given sequence of numbers one by one. （3 分）

```
void insert_min_heap(int item, int *n)
{
   int i;
   i = ++(*n);
   while ((i!=1)&&(item.key<heap[i/2].key)) {
      heap[i] = heap[i/2];
      i /= 2;
   }
   heap[i]= item;
}
```

(2) What is the time complexity of inserting a sequence of n numbers into the minimum heap? （2 分）

(3) Please explain how to adjust the minimum heap after deleting the minimum value from the constructed heap structure of (1). （3 分）

6. The selection problem is described as follows. Given an array A of n distinct numbers and an integer i, with $1 \leq i \leq n$, find the element x that is larger than exactly i-1 other elements in the array A. Below shows the pseudo code of an algorithm RANDOMIZED-SELECT for solving the selection problem.

```
RANDOMIZED-SELECT (A, p, r, i)
1.  if p == r
2.      return A[p]
3.  q = Randomized-Partition (A, p, r)
4.  k = q – p + 1
5.  if i == k
6.      return A[q]
7.  elseif i < k
8.      return RANDOMIZED-SELECT (A, p, q-1, i)
9.  else return RANDOMIZED-SELECT (A, q+1, r, i-k)
```

The function Randomized-Partition (A, p, r) works on $A[p...r]$, randomly picks a pivot element between p and r, and partitions $A[p...r]$ into two sub-arrays $A[p...q]$ and $A[q+1...r]$ such that every element in $A[p...q]$ is less than or equal to every element in $A[q+1...r]$.

(a) What does k in line 4 mean? (5 分)

(b) What is the worst-case running time of RANDOMIZED-SELECT? (5 分)

(c) Suppose we use RANDOMIZED-SELECT to select the smallest element of the set {3, 2, 9, 0, 7}. Describe a sequence of partitions that results in a worst-case performance. (5 分)

7. The rod-cutting problem is described as follows. Given a rod of length n inches and a table of prices p_i for i = 1, 2, …, n, determine the maximum revenue r_n obtainable by cutting up the rod and selling the pieces. An algorithm for solving the rod-cutting problem is shown below.

```
BOTTOM-UP-CUT-ROD(p, n)
1.  let r[0..n] be a new array
2.  r[0] = 0
3.  for j = 1 to n
4.      q = −∞
5.      for i = 1 to j
6.          if q < p[i] + r[j - i])
7.              q = p[i] + r[j - i])
8.      r[j] = q
9.  return r[n]
```

(a) What does line 8 store? (5 分)

(b) Given a rod of 5 inches and the price table below, what is the output of BOTTOM-UP-CUT-ROD? (5 分)

Length	1	2	3	4	5
Price	2	5	6	9	10

(c) Please modify the algorithm to return the decomposition (the best way to cut the rod) as well. (5 分)

8. The Floyd-Warshall algorithm solves the all-pairs shortest-paths problem on a directed graph. The pseudo code is given as follows:

```
Floyd-Warshall(W)
1   n ← rows[W]
2   D^(0) ← W
3   for k ← 1 to n
4       do for i ← 1 to n
5           do for j ← 1 to n
6               d_ij^(k) ← min(d_ij^(k−1), d_ik^(k−1) + d_kj^(k−1))
7   return D^(n)
```

(a) Explain the meaning of $d_{ij}^{(k)}$. (5 分)

(b) Given a graph with edge weight:

$$W = \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix},$$

fill the values of those two "?" elements in $D^{(1)}$. (4 分)

$$D^{(1)} = \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & ? & \square \\ \square & ? & \square & \square \\ \square & \square & \square & \square \end{bmatrix}$$

(c) How can we use the output of the Floyd-Warshall algorithm to detect the presence of a negative-weight cycle? (6 分)

(d) As it appears above, the Floyd-Warshall algorithm requires $\Theta(n^3)$ space, since we compute $d_{ij}^{(k)}$ for $i, j, k = 1, 2, ..., n$. To reduce the space to $\Theta(n^2)$, we simply drop all the superscripts and modify the algorithm as follows:

```
Floyd-Warshall-Modified(W)
1   n ← rows[W]
2   D ← W
3   for k ← 1 to n
4       do for i ← 1 to n
5           do for j ← 1 to n
6               d_ij ← min(d_ij, d_ik + d_kj)
7   return D
```

Does the modified algorithm return correct results? Briefly describe your reasons. (5 分)