國立陽明交通大學
113 學年度碩士班考試入學招生試題

科目：資料結構與演算法(8101)

系所班別：資訊聯招

考試日期：113 年 2 月 2 日 第 1 節

【不可使用計算機】

＊作答前請先核對試題、答案卷(試卷)與准考證之
所組別與考科是否相符！！

【不可使用計算機】＊作答前請先核對試題、答案卷(試卷)與准考證之所組別與考科是否相符！！

選擇題每一題組須全答對才計分。

Part I:Multiple Choice questions(選擇題(單選/多選)請使用答案卡作答)

1. (7%) The following contains the C++ code for a **queue** implemented with a singly linked list.

```
struct Qnode
{
    int key;
    Qnode * next;
    Qnode(int n = 0) : key(n), next(NULL) {}
};
struct Qlist
{
    Qnode * head, * tail;
    Qlist() : head(NULL), tail(NULL) {}
    void push(int n)
    {
        Qnode* node = new Qnode(n);
        if (!head) ____①____ = node;
        if (tail) ____②____ = node;
        tail = ____③____;
    }
    void pop()
    {
        if (head) {
            Qnode* node = ____④____;
            ____⑤____ = ____⑥____;
            delete node;
        }
        if (!head) tail = NULL;
    }
};
```

- ____①____ Which way of filling in the code will make the **push** function work correctly?
  (A) ①: **head** ②: **tail->next** ③: **node**
  (B) ①: **head->next** ②: **tail** ③: **node->next**
  (C) ①: **tail** ②: **tail->next** ③: **head**
  (D) ①: **head->next** ②: **tail** ③: **tail->next**
  (E) None of the above

- ____②____ Which way of filling in the code will make the **pop** function work correctly?
  (A) ④: **tail** ⑤: **tail->next** ⑥: **tail**
  (B) ④: **head** ⑤: **head->next** ⑥: **node->next**
  (C) ④: **tail** ⑤: **tail** ⑥: **node**
  (D) ④: **head** ⑤: **head** ⑥: **head->next**
  (E) None of the above

- ___③___ Based on the code above, given the following statements, what are the key values of the two nodes pointed to by **head** and **tail**?

```
Qlist L = Qlist();
L.pop();  L.push(10);  L.push(5);  L.push(3);  L.pop();
L.push(12);  L.push(8);  L.pop();
```

(A) **head**: 10, **tail**: 8
(B) **head**: 10, **tail**: 12
(C) **head**: 3, **tail**: 8
(D) **head**: 3, **tail**: 10
(E) None of the above

2. (5%) The following code is a C implementation of a sorting algorithm:

```
void FSort(int* a, int left, int right)
{
    if (left < right) {
        static int c = 0;
        int i = left, j = right + 1;
        int d = a[left];
        do {
            do i++; while (a[i] < d);
            do j--; while (a[j] > d);
            if (i < j) {
                int z = a[i]; a[i] = a[j]; a[j] = z;
            }
        } while (i < j);
        int z = a[left]; a[left] = a[j]; a[j] = z;
        printf("c=%d\n", ++c);
        FSort(a, left, j - 1);
        FSort(a, j + 1, right);
    }
}
```

- Assume that we want to use this function to sort a 10-element array m, given by

```
int m[] = { 3,5,9,8,2,0,4,1,6,7 };
```

___④___ Which of the following is the correct call?

(A) `FSort(m, 1, 9)`
(B) `FSort(m, 0, 9)`
(C) `FSort(m, 1, 10)`
(D) `FSort(m, 0, 10)`
(E) None of the above

- ___⑤___ For the above array **m**, assuming that the call above is correct, what is the largest value of **c** in the text output?

(A) 9 (B) 10 (C) 4 (D) 5 (E) 6

- ⑥ Which of the following input array will result in the most number of calls to **FSort**?
  (A) `int m[] = { 3,5,9,8,2,0,4,1,6,7 };`
  (B) `int m[] = { 1,3,5,7,9,0,2,4,6,8 };`
  (C) `int m[] = { 5,6,7,8,9,0,1,2,3,4 };`
  (D) `int m[] = { 0,1,2,3,4,5,6,7,8,9 };`
  (E) None of the above

3. (6%) Insert the keys 80, 69, 99, 16, 73, 30, 41 into a Hash Table of size 13 using **double hashing** with the hash functions h1(k) = k mod 13 and h2(k) = 1 + (k mod 11) for *collision resolution with open addressing.* Based on this scenario, answer the following three questions:

- ⑦ Which index will the key 30 be placed in after resolving any collisions using double hashing?
  (A) 0 (B) 4 (C) 5 (D) 8 (E) None of the above

- ⑧ Which key pair(s) below will cause collision when inserting the given keys to the hash table using the first hash function h1(k)?
  (A) 80 and 69 (B) 16 and 41 (C) 73 and 30 (D) All of the above (E) None of the above

- ⑨ Which index(es) will remain empty after all keys have been inserted?

  (A) 1 (B) 5 and 6 (C) 7 and 9 (D) 10 and 12 (E) None of the above

4. (6%) Given the graph with nodes labeled A to G and weighted edges, apply Kruskal's algorithm to find the minimum spanning tree (MST). The edges and their respective weights are as follows:

Edge	A-B	A-D	B-C	B-D	B-E	C-E	D-E	D-F	E-F	E-G	F-G
Weight	7	5	8	9	7	5	15	6	8	9	11

- ⑩ Which of the following edges are included in the MST after applying Kruskal's algorithm?
  (A) A-D
  (B) C-E
  (C) D-E
  (D) F-G
  (E) None of the above

- ⑪ What is the total weight of the MST for this graph?
  (A) 39
  (B) 40
  (C) 41
  (D) 42
  (E) None of the above
- ⑫ If the edge(s) with the lowest weight in the MST is(are) removed, which node pair(s) become disconnected?
  (A) A and D (B) B and E (C) C and E (D) D and F (E) None of the above

5.（5%） Consider the max-flow problem.

- Let $G = (V, E)$ be a network with edge capacity c_e for all $e \in E$ and source-sink pair $s, t \in V$.

⑬ Which of the following statements is/are true?

(A) If $(S, V - S)$ and $(S', V - S')$ are both valid s-t cuts, then so is $(S \cup S', V - (S \cup S'))$.

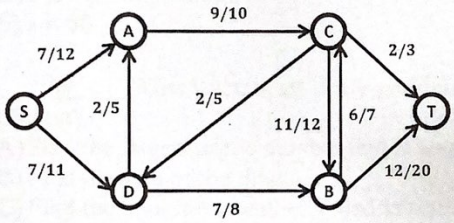
(B) If f_1, f_2 are both feasible s-t flow functions, then so is $f_1 + f_2$, which adds up the flows of each pair of nodes.

(C) For any feasible s-t flow function f with value $Val(f)$ and any feasible s-t cut C with total capacity $c(C)$, we always have $Val(f) \geq c(C)$.

(D) Applying the Bellman-Ford algorithm returns the maximum flow for G.

(E) The Ford-Fulkerson algorithm can be modified to solve the maximum flow problem in polynomial-time.

- Consider the following residual flow network.



⑭ Which of the following statements is/are true?

(A) The path $S \to A \to C \to B \to T$ is a valid augmenting path with value 1.

(B) Let $P = \{S, A, D\}$ and $Q = \{B, C, T\}$. Then (P, Q) is a minimum S-T cut with weight 23.

(C) The Edmonds-Karp algorithm will use $S \to A \to C \to B \to T$ as an augmenting path.

(D) Changing the residual weight settings of the three edges $(D, A), (A, C)$, and (C, D) to $0/5$, $7/10$, and $0/5$ will increase the value of the maximum flow by 2.

(E) If we add a new edge (D, C) with capacity 2 to the network, the value of the minimum S-T cut will also increase by 2.

6. (9%) Consider the following C++ procedure.

```
int sum(long long n)
    {
       int result = 0;
       while (n) {
          result += n % 10;
          n /= 10;
       }
       return result;
    }
```

```
long long foo(long long n, int target)
{
       long long x = n, y = 1;
       while (sum(n) > target) {
          n = n / 10 + 1;
          y *= 10;
       }
       return n * y - x;
    }
```

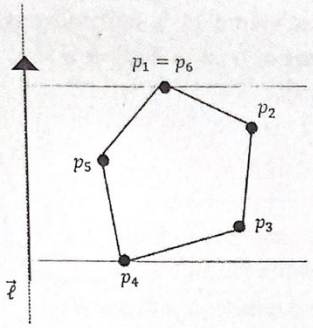
- ___⑮___ Let z be the output of foo(12321, 7). Which of the following statements is/are correct?
  (A) z has 2 digits.
  (B) z % 10 == 9.
  (C) z / 10 == 9.
  (D) z is a prime number.
  (E) z < 50.

- ___⑯___ What is the most likely problem statement with the above C++ procedures as a solution?
  (A) Find the largest prime number that is smaller than target.
  (B) Find the sum of the digits in n.
  (C) Find the minimum number added to n such that the digit sum is at most target.
  (D) Find the maximum number added to n such that the digit sum is at most target.
  (E) Find the minimum number subtracted from n such that the digit sum is at most target.

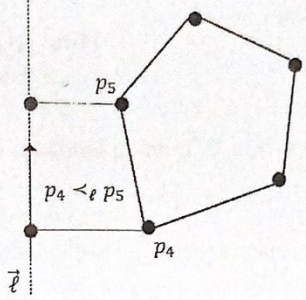
7. (6%) Let p be a prime number and the set $\mathbb{Z}_p = \{0, 1, 2, \cdots, p-1\}$. We can map a vector $a = \langle a_0, a_1, a_2, \cdots, a_{k-1} \rangle$ into a vector $b = \langle b_0, b_1, b_2, \cdots, b_{p-1} \rangle$, where $p > k > 1, a_i, b_i \in \mathbb{Z}_p$ and $b_j = \sum_{i=0}^{k-1} a_i j^i \bmod p$, for $j = 0, .., p-1$.

- ___⑰___ Which of the following statements is/are true?
  (A) Each b_j can be computed in $\Theta(k)$ time.
  (B) Each b_j cannot be computed in $O(k \log p)$ time.
  (C) The vector b can be computed in $O(kp)$ time.
  (D) The mapping from a to b is a 1-1 mapping.
  (E) For any p dimensional vector with entries from $\mathbb{Z}_p$, we can find a corresponding k dimensional vector based on the above mapping.
- ___⑱___ Consider 2 distinct vectors $a = \langle a_0, a_1, a_2, \cdots, a_{k-1} \rangle$, $a' = \langle a'_0, a'_1, a'_2, \cdots, a'_{k-1} \rangle, a_i, a'_i \in \mathbb{Z}_p$, for $i = 0$ to $k-1$. Suppose a is mapped to b and a' to b' as above. Which of the following statements is/are true?
  (A) $b \neq b'$.
  (B) There are at most $(k-1)$ j's such that $b_j = b'_j$.
  (C) Let $c \in \mathbb{Z}_p$. Then $ca + a'$ is mapped to $cb + b'$.
  (D) From b we can reconstruct a.
  (E) If $a = 0$, then $b = 0$.

8. (6%) Consider binary search for finding **extremal points** of a convex polygon **in the given direction**.



(a)



(b)

Let Q be a convex polygon, represented by the vertices of its boundary in clockwise order, say, for example, $Q = p_1, p_2, \ldots, p_n$ with $p_1 = p_n$ and $n \geq 3$, and let $\vec{\ell}$ be a given vector. See Figure (a) for an illustration. Consider the projection of points in Q onto $\vec{\ell}$. For any two points $q, r \in Q$, we write $q \prec_\ell r$, if the projection of r lies further from the starting end of $\vec{\ell}$ than the projection of q. See Figure (b) for an illustration, where p_5 is further than p_4 in the direction of $\vec{\ell}$.

A point $q \in Q$ is said to be maximal with respect to $\vec{\ell}$ if there exists no point $r \in Q$ such that $q \prec_\ell r$. Similarly, $q \in Q$ is minimal with respect to $\vec{\ell}$ if there exists no $r \in Q$ such that $r \prec_\ell q$.

For example, in Figure (a), both p_1 and p_6 are maximal points of Q and p_4 is the minimal point of Q.

- Let $\delta(Q)$ denote the boundary of the convex polygon Q and consider the properties of Q.

⑲ Which of the following statements is/are true?

(A) For any $q \in \delta(Q)$, any vector $\vec{u}$, and any sufficiently small $\epsilon > 0$, exactly one of $q + \epsilon\vec{u}$ and $q - \epsilon\vec{u}$ lies inside Q.

(B) For any $q \in Q \setminus \delta(Q)$, there always exists $r, s \in Q$ with $r \neq s$ and a real number $0 < a < 1$ such that $q = a \cdot r + (1 - a) \cdot s$.

(C) For any $q \in \delta(Q)$, there are an infinite number of lines that tangent Q at q.

(D) For any point q in the plane and any vector $\vec{u}$, if we construct a vector starting from q in the direction of $\vec{u}$, then it always hits $\delta(Q)$ for an even number of times.

(E) For any two points $q, r \in Q \setminus \delta(Q)$, the line segment $\overline{qr}$ is always contained in Q.

- Let a and b be integers with $1 \leq a < b < n$. Consider the points $[a, b] \coloneqq \{p_a, p_{a+1}, \ldots, p_b\}$.

⑳ Which of the following statements is/are true?

(A) If $p_{a+1} \prec_\ell p_a$ and $p_b \prec_\ell p_{b+1}$, then $[a, b]$ contains no maximal points of Q in the direction $\vec{\ell}$.

(B) If $p_a \prec_\ell p_{a+1}$ and $p_b \prec_\ell p_{b+1}$, then no maximal point of Q in the direction $\vec{\ell}$ is contained within $[a, b]$.

(C) If $[a, b]$ contains a minimal point of Q in the direction $\vec{\ell}$, then $p_{b+1} \prec_\ell p_b$ is a possible scenario to happen.

(D) If none of $p_i \prec_\ell p_{i+1}$ and $p_{i+1} \prec_\ell p_i$ holds for some $1 \leq i < n$, then p_i must be either a minimal point or a maximal point of Q in the direction $\vec{\ell}$.

(E) For any $1 < i < n$, the point p_i is a maximal point of Q in the direction $\vec{\ell}$ if and only if $p_{i-1} \prec_\ell p_i$ and $p_i \prec_\ell p_{i+1}$.

- Consider the following sketch of pseudo-code that aims to compute a maximal point of Q for the given direction $\vec{\ell}$ by binary search.
  - $a \leftarrow 0$, $b \leftarrow n$. // to search the range [1,…,n-1]
  - While $a < b - 1$, repeat the following.
    - $c \leftarrow \lfloor (a+b)/2 \rfloor$.
    - Set $b \leftarrow c$ if $[a, c]$ contains a maximal point of Q in the direction $\vec{\ell}$. Otherwise, set $a \leftarrow c$.
  - Output a if it is maximal. Otherwise, output b.

(21) Regarding the above algorithm, which of the following statements is/are true?

(A) When this algorithm terminates, it always outputs a maximal point of Q in the direction $\vec{\ell}$.
(B) For some specific input instance, this algorithm may not terminate at all.
(C) This algorithm may output a point that is not maximal for Q.
(D) This algorithm always terminates on any input instance.
(E) The statement "if $[a, c]$ contains a maximal point of Q" can be tested in $O(1)$ time.

Part II: Non Multiple Choice questions(非選擇題)

1. (13%) There are 5 questions. The following contains two functions for traversing a binary tree:

```
struct Node
{
    char key;
    Node * left, * right;
};

void trv1(Node* root)
{
    std::queue<Node*> Q;
    Q.push(root);
    while (!Q.empty()) {
      Node* cur = Q.front();
      Q.pop();
      printf("%c", cur->key);
      if (cur->left) Q.push(cur->left);
      if (cur->right) Q.push(cur->right);
    }
}
```

```
void trv2(Node* root)
{
  std::stack<Node*> Q;
  Q.push(root);
  while (!Q.empty()) {
    Node* cur = Q.top();
    Q.pop();
    printf("%c", cur->key);
    if (cur->right) Q.push(cur->right);
    if (cur->left) Q.push(cur->left);
    }
}
```

(A) [3%] Use the following two traversal sequences (each character is the key of a node) to reconstruct the binary tree. Note: There are more than one possible trees, and you only need to show one of them.

trv1: OPMRTECUS
trv2: OPRTCSUME

(B) [3%] Start with an empty binary search tree and insert the characters in "**COMPUTERS**" into it. List the resulting preorder traversal sequence.

(C) [2%] Among preorder, postorder, inorder, and level-order traversals, which one(s) can uniquely identify a binary search tree by itself?

(D) [3%] A node of a B-tree is modified from the tree node definition above such that a node can hold up to `M` distinct keys (in an array named **`key`**) and `M+1` pointers to child nodes (in an array named **`child`**), where `M` is a positive integer. For a B-tree node with all its keys and child node pointers being valid, list the requirements of the key values in the subtrees rooted at the nodes pointed to by **`child`** with respect to the key values in **`key`**.

(E) [2%] In B-trees, to minimize the tree depth, each non-leaf node (except for the root) is required to keep at least `K` valid keys. Specify `K` with respect to `M`.

2. (13%) Consider a directed graph G, consisting of four vertices labeled A, B, C, and D. The graph has the following edges with their respective weights: edge from A to B with weight 2, edge from B to D with weight 1, edge from A to C with weight -4, and edge from C to B with weight 3.

Given the Pseudo Code for Bellman-Ford Algorithm:
Input: Graph G with vertices V and edges E, source vertex s
Output: Shortest path from s to other vertices, or detection of a negative cycle

```
1. function BellmanFord(G, s):
2.     // Initialization
3.     for each vertex v in V do
4.         dist[v] :=     (1)
5.         prev[v] := undefined
6.     dist[s] :=          (2)
7.
8.     // Main Loop
9.     for i from 1 to |V|-1:
10.        for each edge (u, v) in E:
11.            if dist[u] + weight(u, v)    (3)    dist[v]:
12.                dist[v] := dist[u] + weight(u, v)
13.                prev[v] := u
14.
15.    // Check for negative weight cycles
16.    for each edge (u, v) in E:
17.        if dist[u] + weight(u, v)    (4)    dist[v]:
18.            return "Graph contains a negative weight cycle"
19.
20.    return dist, prev
```

(A) [4%] What values should be filled in to (1), (2), (3), (4) in the Pseudo Code?
(1) _____ (2) _____ (3) _____ (4) _____

(B) [3%] Let the source vertex be 'A'. After the first iteration of the main loop, what is the updated value of `dist[C]`?
dist[C] := _____

(C) [3%] Let the source vertex be 'A'. In the second iteration of the main loop, after processing the edge from C to B, what will be the new value of `dist[B]`?
dist[B] := _____

(D) [3%] Let the source vertex be 'A'. What will be the final value of `dist[D]` after the algorithm completes its execution?
dist[D] := _____

3. (14%) Recurrence formula is what it takes to solve a problem with the dynamic programming technique. In this problem we examine the recurrence formulas for two different sequence problems.
   - Consider the longest increasing subsequence (LIS) problem. Let $A = a_1, a_2, \ldots, a_n$ be the input sequence of n numbers. For any $0 \le i \le n$, let $T(i)$ denote the maximum length of any **increasing** subsequence of A that **ends** at position i.

   It is well-known that $T(i)$ can be expressed as the following recurrence formula.

$$T(i) = \begin{cases} 0, & \text{if } i = 0, \\ \max\limits_{0 \le j < i,\ a_j < a_i} \{ T(j) + 1 \}, & \text{if } 1 \le i \le n, \end{cases}$$

   where in the formula we define $a_0 := -\infty$ for convenience. The answer to the input sequence is then given by $\max\limits_{1 \le i \le n} T(i)$.
   - Next, consider the "Longest Palindrome Bimodal Subsequence Problem", which is defined as follows. Let $A = a_1, a_2, \ldots, a_n$ be a sequence of n numbers.
     A subsequence $B = b_1, b_2, \ldots, b_m$ of A is called a bimodal subsequence with pivot k, $1 \le k \le m$, if B is an increasing sequence before position k and a decreasing sequence after position k.
     That is, $b_i < b_{i+1}$ holds for all $1 \le i < k$ and $b_j > b_{j+1}$ holds for all $k \le j < m$.
     Furthermore, we say that a bimodal sequence $B = b_1, b_2, \ldots, b_m$ with pivot k is a **palindrome bimodal sequence** if $k = (m+1)/2$.

   For any $1 \le i \le n$, let $PV(i)$ denote the maximum length of any palindrome bimodal subsequence of A that pivots at position i. Write down a valid recurrence formula for $PV(i)$, as the format given above for $T(i)$ for the LIS problem.
   You may define other auxiliary functions necessary to compose the recurrence formula for $PV(i)$. However, make sure you provide a precise and succinct definition.

4. (10%) Consider a fixed positive integer k and a sequence of positive real numbers $a_1, \cdots, a_k, b_1, \cdots b_k$, where $a_i > 0$, $b_i > 1$ for $1 \le i \le k$, such that $\sum_{i=1}^{k} a_i / b_i^r = 1$. (A) [5%] Prove such r exists. (B) [5%] Design an efficient procedure to find the real number r with error within 10^{-6}. (Note: If you don't answer part (A) correctly, then you will not earn any point from this problem!)