

1. (12%) Multiple-choice questions. (單選題)

1.1. (2%) Which of the following is an illegal use of function `put`?

- (a) `cout.put( 'A' );`
- (b) `cout.put( "A" );`
- (c) `cout.put( 'A' ).put( '\n' );`
- (d) `cout.put( 65 );`

1.2. (2%) Assume that `t` is an object of class `Test`, which has member functions `a()`, `b()`, `c()` and `d()`. If the functions `a()`, `b()`, `c()` all return references to an object of class `Test` (using the dereference `this` pointer) and function `d()` returns `void`, which of the following statements will **not** produce a syntax error:

- (a) `t.a().b().d();`
- (b) `a().b().t;`
- (c) `t.d().c();`
- (d) `t.a().t.d();`

1.3. (2%) A function that modifies an array by using pointer arithmetic such as `++ptr` to process every value should have a parameter that is:

- (a) A **non-constant** pointer to **non-constant** data.
- (b) A **non-constant** pointer to **constant** data.
- (c) A **constant** pointer to **non-constant** data.
- (d) A **constant** pointer to **constant** data.

1.4. (2%) A function prototype does **not** have to ____.

- (a) include argument names
- (b) end with a semi-colon
- (c) agree with the function header
- (d) match with all calls of the function

1.5. (2%) Recursion is memory-intensive because:

- (a) Recursive functions tend to declare many local variables.
- (b) Previous function calls are still open when the function call itself and the activation records of these previous calls still occupy space on the call stack.
- (c) Many copies of the function code are created.
- (d) It requires large data values.

1.6. (2%) **static** member functions:

- (a) Can use `this` pointer.
- (b) Can access only other **static** member functions and **static** data members.
- (c) Cannot be called until an object of their class is instantiated.
- (d) Can be declared **const** as well.

2. (3%) Write a C++ function `bool is_palindrome (const char s[ ])` that returns `true` if and only if the C-string `s` reads the same forward and backward (e.g., "mom" and "deed").

3. (4%) What is the output of the following C code?

```
#include <stdio.h>
void main() {
    int i, s;
    for (i = 0, s = 0; i <= 5; ) {
        s += ++i;
    }
    printf("%d\n", s);
}
```

4. (4%) What is the output of the following C code?

```
#include <stdio.h>
void main() {
    printf("%d, %d, %d\n", foo(2), foo(4), foo(6));
}

int foo(int n) {
    int s = 0;
    switch (n) {
        case 1: s = s + n;
        case 2: s = s + 2 * n; break;
        case 3: s = s + 3 * n; break;
        case 4: s = s + 4 * n;
        default: s = s + 5 * n; break;
    }
    return s;
}
```

5. (4%) What is the output of the following C code?

```
#include <stdio.h>
void main() {
    int m = 3, n = 5;
    foo(m, n);
    printf("%d, %d\n", m, n);
}
```

```
void foo(int x, int y) {  
    int z;  
    z = x;  
    x = y;  
    y = z;  
}
```

6. (4%) What is the output of the following C code?

```
#include <stdio.h>  
void main() {  
    int m = 3, n = 5, o = 7;  
    if (m < n || (o = m + 2 >= n)) m = o;  
    printf("%d, %d\n", m, o);  
}
```

7. (4%) What is wrong with the following C code?

```
#include <stdio.h>  
void main() {  
    int m = 3, n = 5, *o;  
    o = goo(m);  
    foo(m, n);  
    printf("%d\n", *o);  
}
```

```
void foo(int x, int y) {  
    int z;  
    z = x;  
    x = y;  
    y = z;  
}
```

```
int * goo(int n) {  
    int z;  
    z = n * n;  
    return &z;  
}
```

8. (15%) Give the output of the following C++ code.

```
#include <iostream>
using namespace std;

class A {
public:
    void print1() { cout << "A1" << endl; }
    virtual void print2() { cout << "A2" << endl; }
    void test() { print1();
                 print2(); }
};

class B: public A {
public:
    void print1() { cout << "B1" << endl; }
};

class C: public B {
public:
    void print2() { cout << "C2" << endl; }
};

int main()
{
    A *p1 = new A;
    B objB;
    B *p2 = &objB;
    C objC;

    p1->test();
    (*p2).test();
    objC.test();

    return 0;
}
```

9. (7%) Suppose A is a sorted array with N elements. If we randomly choose $N/3$ elements of the array and exchange them with their neighbors. What algorithm will be able to sort the resulting array in $O(N)$ time complexity? Explain your answer.
10. (8%)
- (a) (4%) Please describe Open Addressing Hash data structure.
 - (b) (4%) If we need to delete elements from the hash data structure, can we use Open Addressing? Why?
11. (5%) What is the postfix expression of $(A * (C + D)) / (B - E)$?
12. (5%) The matrix chain multiplication problem is to find the optimal way to multiply a chain of n matrices $A_1, A_2, \dots, A_n$ with minimum number of scalar multiplications. Let $d_{i-1} \times d_i$ be the dimension of A_i , and $C(p, q, r)$ be the minimum number of scalar multiplications to multiply two matrices of dimension $p \times q$ and $q \times r$, respectively. Let $m[i, j]$ denote the minimum number of scalar multiplications to multiply sub-chain $A_i, A_{i+1}, \dots, A_j$. Give a recursive formula for $m[i, j]$.
13. (5%) Give a recursive procedure in pseudo code for depth first search on graphs, which returns $d[v]$, $f[v]$ and $\pi[v]$, for every vertex v , where $d[v]$, $f[v]$ and $\pi[v]$ denote the discovery time, finish time and predecessor of v , respectively.
14. (5%) Consider the following function on nonnegative integers m and n .
- $$f(m, n) = \begin{cases} 0 & \text{if } m = 0 \text{ or } n = 0 \\ \max\{f(m-1, n-1) + g(m, n), f(m-1, n), f(m, n-1)\} & \text{otherwise} \end{cases}$$
- Let $G(m, n)$ be an efficient procedure to compute $g(m, n)$. Namely, the running time of $G(m, n)$ is bounded by a polynomial of m and n . Give an efficient procedure $F(m, n)$ to compute $f(m, n)$, using procedure $G(m, n)$.
15. (5%) What is the lower bound of time complexity for any comparison-based sorting algorithm (e.g., bubble sort)? Justify your answer.
16. (5%) The order statistics (finding the i -th smallest element) can be solved by dividing numbers into groups of five, finding the "median of medians of 5-element groups," and recursively solving partitioned subproblems. What is the complexity of a new algorithm which divides numbers into groups of three (instead of five) elements?
17. (5%) The vertex cover problem asks if there exists a set of vertices with size k (called vertex cover) incident to each edge in the graph. Prove the vertex cover problem is NP-complete by reduction from the clique problem.