104 - Introduction to Computer Science - Quiz three

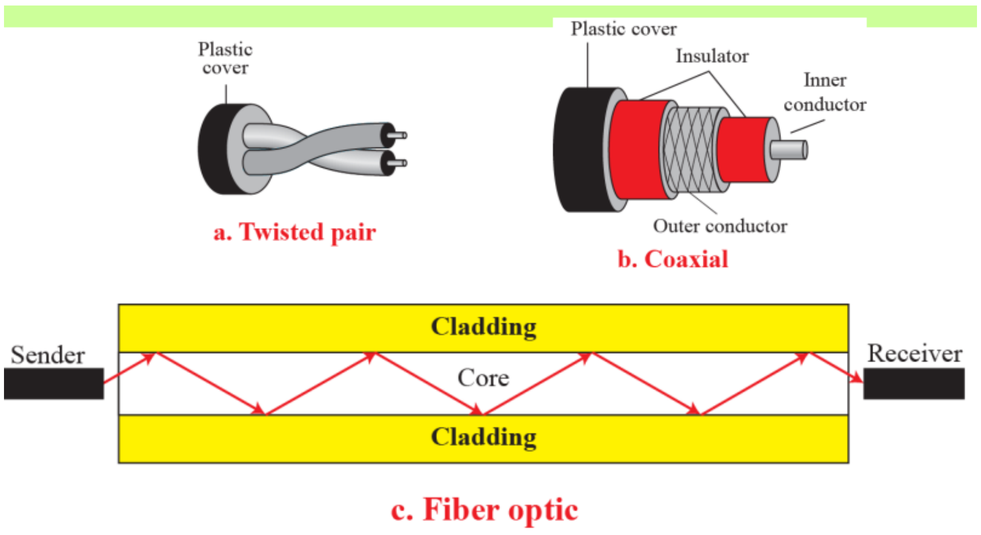
Name：_______________ Student ID：_______________

1. Describe that the comparisons among following transmission media: twisted pair, coaxial, and fiber optic. (5%)

   A twisted pair consists of two conductors (normally copper), each with its own plastic insulation, twisted together. One of the wires is used to carry signals to the receiver, and the other is used only as a ground reference.

   Coax has a central core conductor of solid or stranded wire (usually copper) enclosed in an insulating sheath, which is, in turn, encased in an outer conductor of metal foil, braid, or a combination of the two.

   The fiber-optic cable is made of glass or plastic and transmits signals in the form of light. This technology uses the property of a beam of light that is either refracted (come back) when encounter a medium of less dense. Covering a glass or plastic medium by another less dense medium (called cladding) guides the light through the medium.



2. How is paging different from partitioning? (5%)

   In partitioning, memory is divided into variable-length sections, each of which holds one complete program. In paging, memory is divided into much smaller fixed-length sections as is the program itself; the program does not have to be contiguous in memory.

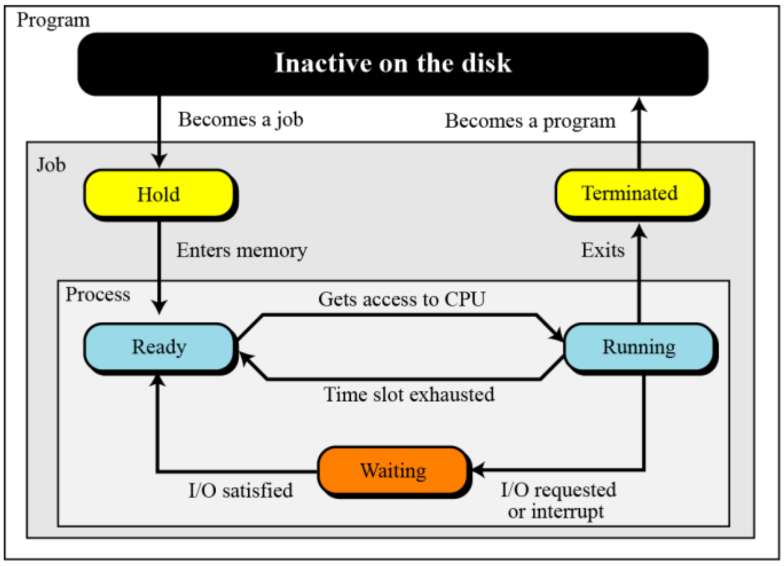
3. What is the difference between deadlock and starvation? (5%)

   Deadlock happens when processes are all waiting for resources held by other processes: they are all waiting for each other. This happens when the operating system does not put resource restrictions on processes. Starvation happens when the operating system puts too many resource restrictions on a process. If a process must wait until it can get all of the resources that it needs before it starts to execute, it may never start.

4. An operation system uses virtual memory but requires the whole program to be in physical memory during execution (no paging or segmentation). The size of physical memory is 100 MB. The size of virtual memory is 1 GB. How many of them can be in memory at any time? How many of them must be on disk? (10%)
   Case one: 考慮實體記憶體不包含虛擬記憶體 and 考慮是整體的記憶體 and 考慮 1 GB 為 1000MB →
   Total memory = 1000 + 100 = 1100 MB.
   Number of program = 1100 / 10 = 110. (In entire memory)
   Number of program = 1000 / 10 = 100. (On disk)
   Case two: 考慮實體記憶體包含虛擬記憶體 and 考慮是整體的記憶體 and 考慮 1 GB 為 1000MB
   Number of program = 1000 / 10 = 100. (In entire memory)
   Number of program = (1000-100) / 10 = 90. (On disk)
   Case three: 考慮實體記憶體不包含虛擬記憶體 and 只考慮在實體記憶體 and 考慮 1GB 為 1000MB
   Number of program = 100 / 10 = 10. (In physical memory)
   Number of program = (1000) / 10 = 100. (On disk)
   Case four: 考慮實體記憶體包含虛擬記憶體 and 只考慮在實體記憶體 and 考慮 1GB 為 1000MB
   Number of program = 100 / 10 = 10. (In physical memory)
   Number of program = (1000-100) / 10 = 90. (On disk)
   ※考慮 1GB 為 1024MB 則 1024/10=102※

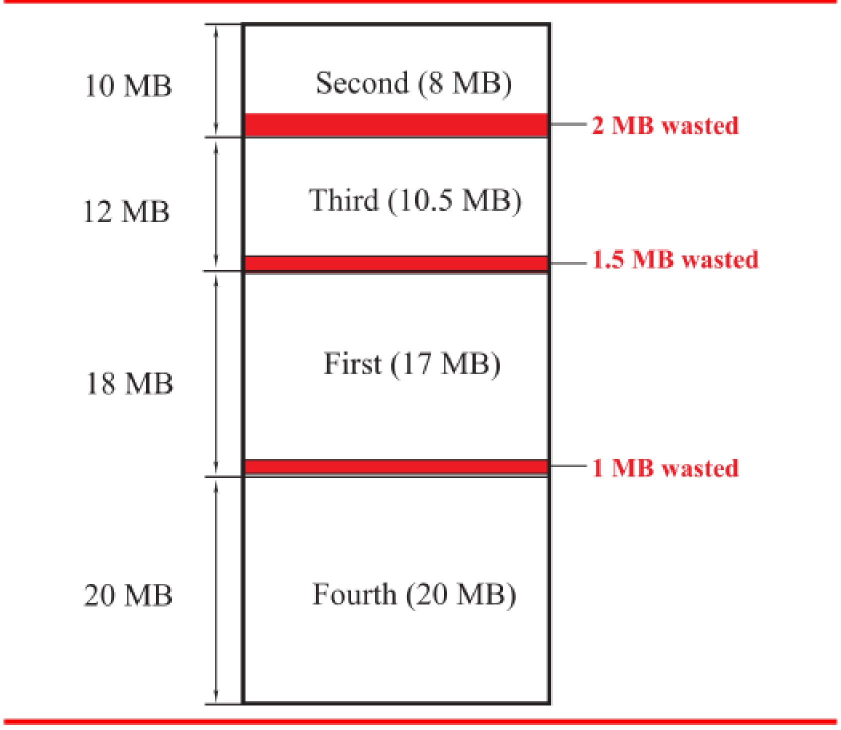
5. What is the status of a process in each of the following situations? (10%)
   a. The process is using the CPU.
   b. The process has finished printing and needs the attention of the CPU again.
   c. The process has been stopped because its time slot is over.
   d. The process is reading data from the keyboard.
   e. The process is printing data.
   a. Running
   b. Ready
   c. Ready
   d. Waiting

e. Waiting



6. A multiprogramming operation system uses an apportioning scheme and divides the 60 MB of available memory into four partitions of 10 MB, 12 MB, 18 MB, and 20MB. The first program to be run needs 17 MB and occupies the third partition. The second program needs 8 MB and occupies the first partition. The third program needs 10.5 MB and occupies the second partition. Finally, the fourth program needs 20 MB and occupies the fourth partition. What is the total memory used? And is the total memory wasted? What percentage of memory is wasted? (10%)

Figure P7-4 *Partitioning with different memory need*



Total memory used = 17 + 8 + 10.5 + 20 = 55.5 MB.

Total memory wasted = 2 + 1.5 +1 = 4.5 MB.

Percent memory wasted = 4.5 / 60 × 100 = 7.5%.

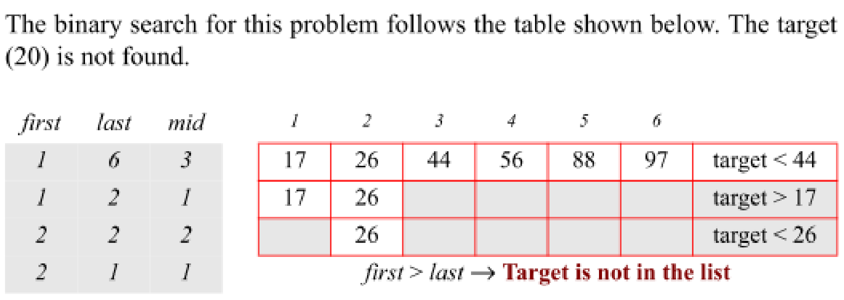
7. Using the insertion sort algorithm, manually sort the following list and show your work in each pass using a table. (10%)

14 7 23 31 40 56 78 9 2

Pass	List								
	14	7	23	31	40	56	78	9	2
1	14	7	23	31	40	56	78	9	2
2	7	14	23	31	40	56	78	9	2
3	7	14	23	31	40	56	78	9	2
4	7	14	23	31	40	56	78	9	2
5	7	14	23	31	40	56	78	9	2
6	7	14	23	31	40	56	78	9	2
7	7	14	23	31	40	56	78	9	2
8	7	9	14	23	31	40	56	78	2
9	2	7	9	14	23	31	40	56	78

8. A list contains the following elements. Using the binary search algorithm, trace the steps followed to find 20. At each step, show the values of *first*, *last*, and *mid*. (10%)

17 26 44 56 88 97



The binary search for this problem follows the table shown below. The target (20) is not found.

first	*last*	*mid*
1	*6*	*3*
1	*2*	*1*
2	*2*	*2*
2	*1*	*1*

1	*2*	*3*	*4*	*5*	*6*	
17	26	44	56	88	97	target < 44
17	26					target > 17
	26					target < 26

first > *last* ⟶ **Target is not in the list**

9. Write an algorithm in pseudocode for the selection sort using a subalgorithm to find the smallest integer in the unsorted sublist. (10%)

Algorithm P8-31a *Selection sort algorithm calling smallest subroutine*

```
Algorithm: SelectionSort(list, n)
Purpose: to sort a list using selection sort method
Pre: A list of numbers
Post: None
Return:
{
    wall ← 1                                   // Set wall at the left of first element
    while (wall < n)
    {
        smallest ← FindSmallest (list, wall, n)   // Call FindSmallest
        Temp ← $A_{wall}$        // The next three lines perform swapping
        $A_{wall}$ ← $A_{smallest}$
        $A_{smallest}$ ← Temp
        wall ← wall + 1          // Move wall one element to the right
    }
    return SortedList
}
```

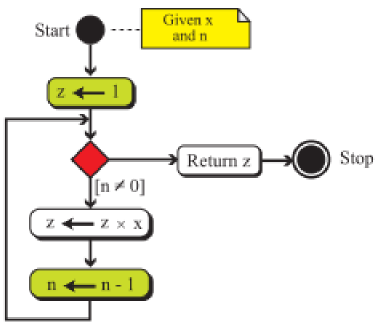
Algorithm P8-31b *Smallest subroutine*

```
Algorithm: FindSmallest(list, wall, n)
Purpose: To find the smallest number in an unsorted list
Pre: A list of numbers
Post: None
Return: The location of the smallest element in the unsorted list
{
    smallest ← wall        // Assume the first element is the smallest one
    cur ← wall             // The current item is the one left to the wall
    while (cur < n)
    {
        if ($A_{cur} < A_{smallest}$)
            smallest ← cur
        cur ← cur + 1                  // Move the current element
    }
    return smallest
}
```

10. Using the UML diagram for the product algorithm, draw a diagram to calculate the value of x^n, when x and n are two given integer. (10%)

Figure 8-38 shows the UML for finding the power of an integer with an integral exponent.

Figure P8-38 *Power*



11. What is the difference between iterative and recursive? Please give an example. (5%)
    Recursion is a process in which an algorithm calls itself.
    The simple example is that we consider the calculation of a factorial.

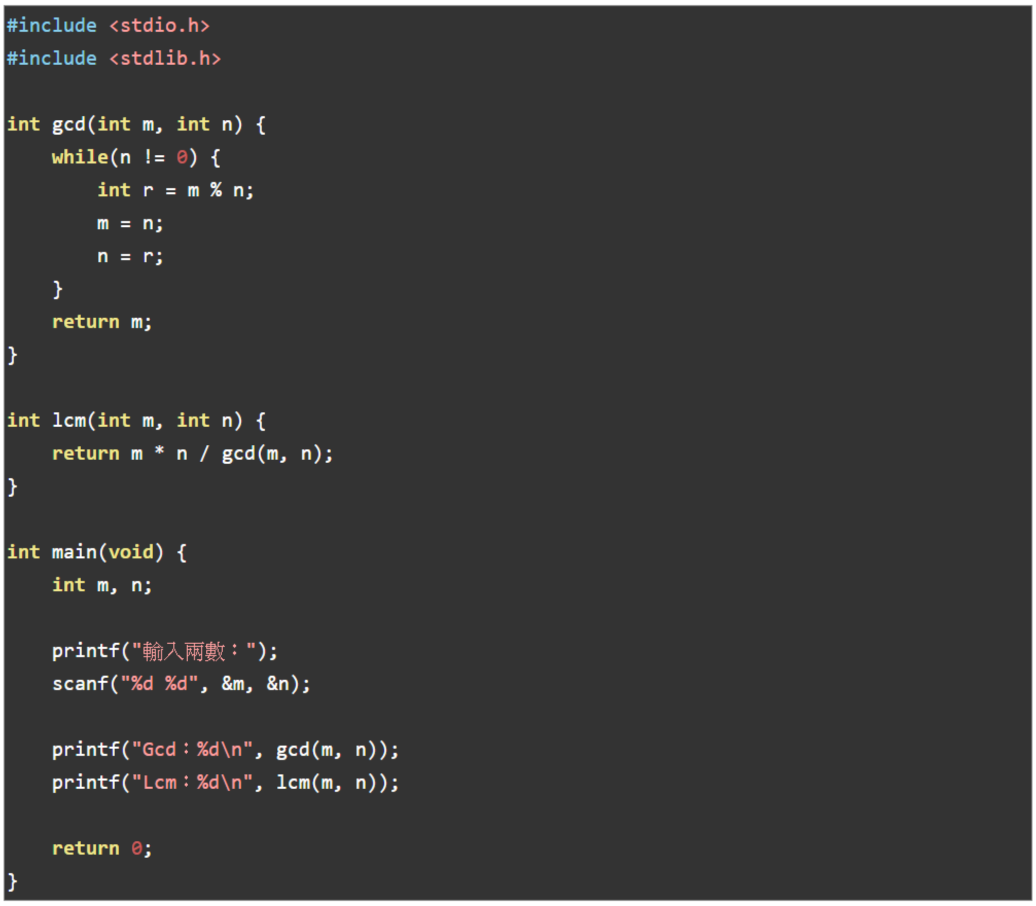
$$\text{Factorial}(n) = \begin{bmatrix} 1 & & & \text{if } n = 0 \\ n \times (n-1) \times (n-2) & \cdots & 3 \times 2 \times 1 & \text{if } n > 0 \end{bmatrix}$$

Figure 8.21 Iterative definition of factorial

$$\text{Factorial}(n) = \begin{bmatrix} 1 & \text{if } n = 0 \\ n \times \text{Factorial}(n-1) & \text{if } n > 0 \end{bmatrix}$$

Figure 8.22 Recursive definition of factorial

12. Write a program in C language to find the greatest common divisor (GCD) and least common multiple (LCM) of two integers *a* and *b*. (10%)
    Please turn *m* to *a* and *n* to *b* in following program.



```c
#include <stdio.h>
#include <stdlib.h>

int gcd(int m, int n) {
    while(n != 0) {
        int r = m % n;
        m = n;
        n = r;
    }
    return m;
}

int lcm(int m, int n) {
    return m * n / gcd(m, n);
}

int main(void) {
    int m, n;

    printf("輸入兩數：");
    scanf("%d %d", &m, &n);

    printf("Gcd：%d\n", gcd(m, n));
    printf("Lcm：%d\n", lcm(m, n));

    return 0;
}
```