基礎程式設計（Java）　答案卷

隨堂練習卷

建議作答 40 分鐘 · 總分 100

§ 1　一、選擇題（每題 6 分，共 30 分）

1. **(B)** — 每個遞迴方法都要有基底條件（終止點）與遞迴步驟（縮小後呼叫自己），缺一不可。 2. **(B)** — 基底條件是最簡單、可直接回答、不再呼叫自己的情況，是遞迴的終止點；可以有多個（如費氏有兩個）。 3. **(C)** — 漏基底條件或不收斂 → 無限遞迴，堆疊框疊到超過上限丟 `StackOverflowError`（編譯期看不出來）。 4. **(B)** — 遞迴沿用一般的呼叫堆疊：每次呼叫 push 一個活動記錄、到基底後依序 pop（LIFO），最後 push 的最先 pop；(D) 方向講反、(C) 各框有自己的參數。 5. **(C)** — 遞迴每次呼叫都多一個堆疊框（記憶體＋時間開銷），迭代用迴圈不需；兩者可互換、遞迴不一定較快。

§ 2　二、遞迴展開與堆疊（共 16 分）

6. 展開與回溯：

```
展開（往下 push 到基底）：
  factorial(4) = 4 * factorial(3)
  factorial(3) = 3 * factorial(2)
  factorial(2) = 2 * factorial(1)
  factorial(1) = 1            ← 基底條件

回溯（往上 pop、回傳值往回乘）：
  factorial(1) = 1
  factorial(2) = 2 * 1 = 2
  factorial(3) = 3 * 2 = 6
  factorial(4) = 4 * 6 = 24
```

填空：`factorial(1)` ＝ **1**；`factorial(2)` ＝ **2**、`factorial(3)` ＝ **6**、`factorial(4)` ＝ **24**。 7. 輸出：

```
3
```

費氏數列 `0, 1, 1, 2, 3…`，`fib(4) = 3`。`fib(4)` 共呼叫 `fib` **9** 次：`fib(4)` 拆成 `fib(3)+fib(2)`；`fib(3)` 子樹 5 次（`fib(3)`、`fib(2)`、`fib(1)`、`fib(1)`、`fib(0)`），`fib(2)`

子樹 3 次（`fib(2)`、`fib(1)`、`fib(0)`），加上 `fib(4)` 自己：5 + 3 + 1 = 9。（同一子問題被重算，正是費氏遞迴效能指數成長的原因。）

§ 3 三、改錯（每題 6 分，共 18 分）

8. 漏基底條件，`n` 一路遞減成負數仍不停 → 無限遞迴、`StackOverflowError`。補上終止點：

```
static long fact(int n){
    if (n <= 1) return 1;            // 基底條件
    else return n * fact(n - 1);
}
```

9. 遞迴步驟 `sum(n)` 沒讓引數變小，永遠到不了基底條件 → 無限遞迴。應呼叫 `sum(n - 1)`：

```
static int sum(int n){
    if (n <= 0) return 0;
    else return n + sum(n - 1);      // 引數減 1，收斂到 n<=0
}
```

10. 輾轉相除法的基底是 `b == 0` 回 `a`、遞迴是 `gcd(b, a % b)`，題目把兩者寫反了。改正：

```
static int gcd(int a, int b){
    if (b == 0) return a;            // 基底條件
    else return gcd(b, a % b);       // 餘數縮小，收斂到 b==0
}
```

§ 4 四、程式設計（共 36 分）

11.（18 分）

```
public class SumRecursion {
    public static int sum(int n) {
        if (n <= 0) return 0;              // 基底條件
        else return n + sum(n - 1);        // 遞迴步驟：引數減 1
    }
    public static void main(String[] args) {
        System.out.println("sum(10) = " + sum(10));   // 55
    }
}
```

（評分：基底條件 6 分、遞迴步驟收斂且正確相加 8 分、`main` 印 `sum(10)` 4 分。用 `n == 1` 回 1 當基底亦可。） 12.（18 分）

```java
public class HanoiTowers {
    public static void hanoi(int n, char from, char to, char aux) {
        if (n == 1) {
            System.out.println("搬 1：" + from + " → " + to);   // 基底條件
        } else {
            hanoi(n - 1, from, aux, to);                          // ① n-1 片 → 輔助柱
            System.out.println("搬 " + n + "：" + from + " → " + to);  // ② 最大片 → 目標
            hanoi(n - 1, aux, to, from);                          // ③ n-1 片 → 目標
        }
    }
    public static void main(String[] args) {
        hanoi(3, 'A', 'C', 'B');     // 共 2^3 - 1 = 7 步
    }
}
```

（評分：基底條件 4 分、三步遞迴順序正確 10 分、`main` 正確呼叫 4 分。輸出共 7 行搬移步驟。）