Question 1

```
public static int m1(int[] a) {
    int n = a.length, total = 0;        C1
    for (int j = 0; j < n; j++)         C2 * n
        total = total + a[j];           C3 * n
    return total;                       C4
}
```

$$
\begin{aligned}
T(n) &= C_1 + C_2(n) + C_3(n) + C_4 \\
&= n(C_2 + C_3) + (C_1 + C_4) \\
&= n(2C_x) + 2C_x
\end{aligned}
$$

By ignoring the constants we get:

$$T(n) = n$$

Therefore, the upper bound is:

$$O(n)$$

Question 2

```
public static int m2(int[] a) {
    int n = a.length, total = 0;        C1
    for (int j = 0; j < n; j += 2)      C2 * Ceiling(n/2)
        total = total + a[j];           C3 * Ceiling(n/2)
    return total;                       C4
}
```

- When $n = 3$, the for-loop runs $\lceil 3/2 \rceil = \lceil 1.5 \rceil = 2$ times.
- When $n = 4$, the for-loop runs $\lceil 4/2 \rceil = \lceil 2 \rceil = 2$ times.
- When $n = 5$, the for-loop runs $\lceil 5/2 \rceil = \lceil 2.5 \rceil = 3$ times.

$\vdots$

$$
\begin{aligned}
T(n) &= C_1 + C_2\lceil \frac{n}{2} \rceil + C_3\lceil \frac{n}{2} \rceil + C_4 \\
&= \lceil \frac{n}{2} \rceil (C_2 + C_3) + (C_1 + C_4) \\
&= \lceil \frac{n}{2} \rceil (2C_x) + 2C_x
\end{aligned}
$$

By ignoring the constants we get:

$$T(n) = \lceil \frac{n}{2} \rceil$$

Therefore, the upper bound is $O(n/2)$. But $O(n/2)$ is not valid, and to be on the safe side, we can conclude that the upper bound is:

$$O(n)$$

Question 3

```
public static int m3(int[] a) {
    int n = a.length, total = 0;              C1
    for (int j = 0; j < n; j++)               C2 * n
        for (int k = 0; k <= j; k++)          C3 * (SUM j = 0 to n - 1)(j + 1)
            total = total + a[j];             C4 * (SUM j = 0 to n - 1)(j + 1)
    return total;                             C5
}
```

We only need to focus on how many times the inner-loop runs. For the inner-loop, we have:

$$T(n) = \sum_{j=0}^{n-1}(j+1), (n-1) \geq 0$$

Which is equivalent to:

$$T(n) = \sum_{j=1}^{n}(j), n \geq 1$$

This summation series can be transformed into:

$$T(n) = n(n+1)/2 = \frac{n^2}{2} + \frac{n}{2}$$

To test that these equations are indeed equivalent:

$$\begin{aligned}
T(3) &= \sum_{j=0}^{n-1}(j+1) = \sum_{j=0}^{3-1}(j+1) = 1+2+3 = 6 \\
&= \sum_{j=1}^{n}(j) = \sum_{j=1}^{3}(j) = 1+2+3 = 6 \\
&= \frac{n^2}{2} + \frac{n}{2} = \frac{3(3+1)}{2} = \frac{12}{2} = 6 \\
T(4) &= \sum_{j=0}^{n-1}(j+1) = \sum_{j=0}^{4-1}(j+1) = 1+2+3+4 = 10 \\
&= \sum_{j=1}^{n}(j) = \sum_{j=1}^{4}(j) = 1+2+3+4 = 10 \\
&= \frac{n^2}{2} + \frac{n}{2} = \frac{4(4+1)}{2} = \frac{20}{2} = 10
\end{aligned}$$

Therefore:

$$T(n) = \frac{n^2}{2} + \frac{n}{2}$$

By ignoring the insignificant terms we have:

$$T(n) = \frac{n^2}{2}$$

By ignoring the constants we have:

$$T(n) = n^2$$

In conclusion, the upper bound is:

$$O(n^2)$$

Question 4

```
public static int m4(int[] a, int[] b) {
    int n = a.length, total = 0;                C1
    for (int i = 0; i < n; i++) {               C2 * n
        int total = 0;                          C3 * n
        for (int j = 0; j < n; j++)             C4 * n^2
            for (int k = 0; k <= j; k++)        C5 * n * (SUM j = 0 to n)(j)
                total = total + a[j];           C5 * n * (SUM j = 0 to n)(j)
        if (b[i] == total)                      C6 * n
            count++;                            C7 * C, (C <= n)
    }
    return count;                               C8
}
```

Same logic as in question 3, we only need to focus on the innermost-loop. We already proved in the previous question that for the innermost-loop with k, if there is only one outer-loop enclosing it (loop with j), it's running time is $T(n) = \sum_{j=1}^{n}(j) = \frac{n^2}{2} + \frac{n}{2}$. Thus, by adding another outer-loop (loop with i from 0 to n in this case), we only need to multiply the running time we get from the previous question by n. Therefore, the running time of the innermost-loop in this question is:

$$T(n) = n \times \sum_{j=1}^{n}(j) = \frac{n^3}{2} + \frac{n^2}{2}$$

By ignoring the insignificant terms and the constants, we can conclude that the upper bound is:

$$O(n^3)$$

(1)
Number Theory: The product of two odd integers is odd.

Proof: Let x and y be odd in integers.

- $x = 2m + 1$
- $y = 2n + 1$

Assumption: x * y is even ($x*y = 2k$)

$$
\begin{aligned}
(2m + 1)(2n + 1) &= 4mn + 2m + 2n + 1 \\
&= 2(2mn + m + n) + 1 \\
&= 2k + 1 = 2k
\end{aligned}
$$

Contradiction: -
Therefore, the product of two odd integers is not even.

(2)
Proof by mathematical induction: $1 + 2 + 2^2 + ... + 2^n = 2^{n+1} - 1$

Basis: n = 1

LHS: -	RHS: -
1 + 2 = 3	$2^{n+1} - 1$ $= 2^{1+1} - 1$ = 3
LHS = RHS	

Induction: Assume true n = k $\quad 1 + 2 + 2^2 + ... + 2^k = 2^{k+1} - 1$
Showing P(n) is true for n=k+1)
$1 + 2 + 2^2 + ... + 2^k + 2^{k+1} = 2^{(k+1)+1} - 1$

LHS: -
$P(k) - 1 + 2^{k+1}$
$= 2^{k+1} - 1 + 2^{k+1}$
$= 2^{1+k+1} - 1$

- LHS = RHS, therefore p(k+1) is true.