1)

```
num=0;
for (i = n; i>= 0; i−−)
        num−−;
```

Answer: Running time is O(n)
Explanation: after the 1st instruction, there is a single loop that runs (n+1) times; each time the loop runs it executes the instruction in the loop header and 1 instruction in the body of the loop. The total number of instructions is 2*(n+1) +1 (for the last loop check) + 1 = 2n + 4 = O(n).

2)

```
num=0;
for (i = 0; i<= n ∗ n; i=i+2)
        num=num+2;
```

Answer: Running time is
Explanation: after the 1st instruction, there is a single loop that runs (n^2)/2 + 2 (including last loop check);
therefore 2[(n^2)/2 + 2] + 1 = n^2 + 5 = O(n^2).

0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36

n=2	4
n=3	6
n=4	10
n=5	14
n=6	20

3)

```
num=0;
for (i = 1; i<= n; i=i*2)
        num++;
```

Answer: Since the number of iterations decreases by half, loop has logN +2 complexity (inclusive of last loop check); therefore 2(logN + 2) +1 = 2logN + 5 = O(logN).

4)

```
num=1;
for (i = 0; i<n; i++)
        for (j = 0; j<=i; j++)
                num = num * 2
```

Answer: O(n^2)

5)

```
p=10;
num=0;
plimit=100,000;
```

```
for (i = p; i<=plimit; i++)                    ((10^5) – 9) n
        for (j = 1; j<=i; j++)                 ((10^5) – 9) n
                num = num + 1;
```

Answer: O(n^2)

6)

```
num=0;

for (i = n*n; i>=0; i=i/2)
        for (j = 1; j<=n; j++)
                num = num + i;
```

Answer: O(n^3 logN)

7)

```
num=0;
for (i = 0; i<n*n-1; i++)
        for (j = 0; j< i ∗ i; j++)
                num = 2*num;
```

Answer:

8)

```
num=0;
for (i = 0; i<= n*n; i++)
        num++;
for (i = 1; i<=n; i=i*2)
        for (j=n; j>= 1; j=j/2)
                num++;
```

Ans: n^2 + logn*logn

9)

```
for (i = 0; i < n; i++) {
        smallest = i;
        for (j = i+1; j <= n; j++) {
                if (a[j] < a[smallest])
                        smallest = j;
        }
        swap(a, i, smallest); // has three instructions
}
```

Ans: O(n^2)

10)

```
num = 0;
i = 0;

while (i<n) {
    j = 0;
    while (j<100) {
        //constant time operations
        j++;
    }
    i++;
}
```

Ans: O(100n^2)