1) Sumar todos los grupos de 3 de un arreglo

```
// Function to compute the sum of all triplets in the array
void sumOfAllTriplets(int arr[], int n) {
  // Iterate over the first element of the triplet
  for (int i = 0; i < n; i++) {
    // Iterate over the second element of the triplet
    for (int j = i + 1; j < n; j++) {
      // Iterate over the third element of the triplet
      for (int k = j + 1; k < n; k++) {
        // Compute the sum of the current triplet
        int sum = arr[i] + arr[j] + arr[k];
        // Print the sum of the current triplet
        cout << "Sum of (" << arr[i] << ", " << arr[j] << ", " << arr[k] << ") is " << sum << endl;
      }
    }
  }
}
```

A) O(N)
B) O(N^2)
C) O(N^3)
D) O(log(N))

2) Sumar todos los pares de un arreglo:

```
// Function to compute the sum of all pairs in the array
void sumOfAllPairs(int arr[], int n) {
  int sum = 0;
  // Outer loop goes through each element
  for (int i = 0; i < n; i++) {
    // Inner loop adds the current element of the outer loop with each element of the array
    for (int j = 0; j < n; j++) {
      sum += arr[i] + arr[j];
    }
  }
  // Print the total sum of all pairs
  cout << "Sum of all pairs: " << sum << endl;
}
```

A) O(N)
B) O(N^2)
C) O(N^3)
D) O(log(N))

3) Multiplicar matriz

```
void matrixMultiply(int A[N][N], int B[N][N], int C[N][N]) {
  for (int i = 0; i < N; i++) {
```

```
    for (int j = 0; j < N; j++) {
      C[i][j] = 0;
    }
  }

  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
      for (int k = 0; k < N; k++) {
        C[i][j] += A[i][k] * B[k][j];
      }
    }
  }
}
```

A) O(N)
B) O(N^2)
C) O(N^3)
D) O(log(N))

4) Búsqueda Binaria

```
int binarySearch(int arr[], int l, int r, int x) {
  while (l <= r) {
    int m = l + (r - l) / 2; // Calculate the middle index

    // Check if x is present at mid
    if (arr[m] == x) {
      return m; // x found at index m
    }

    // If x greater, ignore left half
    if (arr[m] < x) {
      l = m + 1;
    }
      // If x is smaller, ignore right half
    else {
      r = m - 1;
    }
  }

  // If we reach here, then the element was not present
  return -1;
}
```

A) O(N)
B) O(N^2)
C) O(N^3)
D) O(log(N))

5) Ordenamiento Burbuja

```
// Function to perform Bubble Sort
void bubbleSort(int arr[], int n) {
  // Variable to keep track of whether a swap occurred in the inner loop
  bool swapped;

  // Outer loop for each pass
  for (int i = 0; i < n - 1; i++) {
    // Initially, no swaps have occurred on this pass
    swapped = false;

    // Inner loop for comparing adjacent elements
    for (int j = 0; j < n - i - 1; j++) {
      // Compare adjacent elements
      if (arr[j] > arr[j + 1]) {
        // Swap if elements are in wrong order
        swap(arr[j], arr[j + 1]);
        // Set swapped to true indicating a swap occurred
        swapped = true;
      }
    }

    // If no swaps occurred in the inner loop, the array is sorted
    if (!swapped)
      break;
  }
}
```

A) O(N)
B) O(N^2)
C) O(N^3)
D) O(log(N))

6)

```
int a = 0, i = N;
while (i > 0) {
  a += i;
  i /= 2;
}
```

A) O(N)
B) O(sqrt(N))
C) O(N/2)
D) O(log(N))