**Q1.**

double sum\_triples(double array[], int n) {//n: size of the array. Assume n is divisible by 3

  double sum=0; // 1 time

  for (int i=0; i<n; i=i+3) // times

    sum = sum + array[ i ];   // times

  return sum; // 1 time

}

**Q2.**

double sum\_exponentials(int n){        //n is a power of 3, i.e., n=3^k or k=log n base 3

int sum=0; // 1 time

for (int i=1; i<n; i=i\*3)  // times

      sum = sum + i; // times

return sum; // 1 time

}

**Q3.**

for (int i=0; i<n; i++) { // n+1 times

    for (int j=n; j>=i; j--) // times =

         cout << i << “,” << j <<endl; // times =

 }

**Q4.**

//assume n is divisible by 2 => n = 2\*k

for (int i=0; i<n; i++) { // n+1 times

  for (j=n/2; j>i; j--) // times =

       sum = i+j; // times =

}

**Q5.**

//matrix multiplication of A[m][n] and B[n][p]. The product is saved into C[m][p].

void mult\_matricies( double A[][n], double B[][p], double C[][p], int m, int n , int p ){

for (int i=0; i<m; i++) { // m +1 times

    for (int j=0; j<p; j++){ //m(p+1) times

          C[i][j] = 0; // m(p) times

           for ( int k=0; k<n; k++) { // m(p)(n+1) times

                   C[i][j] += A[i][k] \* B[k][j]; // m(p)(n) times

            }//for-k

     }//for-j

}//for-i

}