First we aim to formulate the problem in mathematical terms. For this we enumerate the n vertices of the regular polyhedron in clockwise direction by the set

$$S_n := \{0, 1, \cdots, n-1\}$$

Moreover we equip this set with the group structure of $\mathbb{Z}_n$, that is,

$$a + b := (a + b) \mod n$$

where the '+'-operation on the r.h.s is the one from $\mathbb{Z}$. In words, adding 1 to a vertex results in the next vertex in clockwise direction.

For every pair (i, j) with $i, j \in S_n$ there exists a unique $k < n$ such that $i + k = j$. This can be used to split S_n into two disjoint sets:

$$A := \{i + s \ : \ 0 \leq s \leq k\}$$

$$B := S_n \backslash A$$

In the sequel these sets will be just referred as the A-set resp. B-set of (i, j).

We call a pair $(i, j) \in S_n \times S_n$ a **diagonal**, if for some $1 < k < n - 1$ $i + k = j$. Such a pair corresponds to a diagonal in the considered regular polyhedron.

Let (i, j) be a diagonal and A, B the disjoint sets defined above. Then a second diagonal (k, l) is said to **intersect** with (i, j) if either $k \in A \ \wedge \ l \in B$ or $l \in A \ \wedge \ k \in B$. One easily verifies that this definition exactly describes two diagonals to intersect in the inner of the polyhedron.

This definitions allow us to formulate the problem in mathematical terms:

Problem. *For given $n, k \in \mathbb{Z}_+$ with $n > k$ find the number of all k-element sets of non-intersecting diagonals in S_n.*

The following algorithm (pseudo-code) is solving the above problem:

Algorithm.

```
res <- 0
for i in [0, ..., n-1]
  for j in [i+2, ..., i-2]
    if k = 1 and j >= i // (*)
      res <- res + 1
    else
      res <- res
        + count_sub_diags(k - 1, i, j, n, i)


def count_sub_diags(k, lower, upper, n, root)
  if upper + 1 = lower or upper + 2 = lower
    return 0

  if k = 1
    return 1

  res <- 0
  for i in [upper + 2, ..., lower - 1]
    if i >= root // (**)
      res <- res
        + count_sub_diags(k - 1, lower, i, n, root)

  return res
```

Note, the intervals $[i, ..., j]$ used in the above algorithm are meant to present cyclic intervals. So for instance if $n = 5$, the interval $[3, ..., 2]$ presents the tuple $(3, 4, 0, 2)$. In addition, operations on vertexes are to be interpreted as $\mathbb{Z}_n$-group operations. So, for instance, $i + 1$ in terms of integers means: $(i + 1) \mod n$.

The algorithm starts by using $i = 0$ as the lower end of a diagonal and $j = 2$ as its upper end. In case of $k = 1$, we found a k-element

set of diagonals and we can increase the counter res. Otherwise, we increase the counter by the number of $(k-1)$-element sets of diagonals found in the B-set of (i, j). This number is computed in the method $count\_sub\_diags$.
This is proceeded by first iterating j, that is, the upper end of the diagonal, until it reaches two vertexes before i, and then iterating i, that is the lower end of the diagonal.

Theorem. *The above algorithm correctly returns the number of all k-element sets of diagonals.*

Proof. The proof will be done by induction on n. It is easy to see, that the algorithm correctly works for the case $n \leq 4$. Next, we show that if it correctly works for $n-1$ then it will work for n as well.
To show the algorithm to collect all k-element sets of diagonals we consider a given set S of k non intersecting diagonals. Let $(i, j) \in S$ be any of these diagonals and D then one within the cyclic interval $[i, i+1, \cdots, j]$ with starting point most next to i and containing no diagonals of S inside its A-set. It is clear such a diagonal to exist by the assumption all diagonals in S to not intersect. Now, let the vertex i in the first for-loop be the one corresponding to the lower end of D. Since all the remaining $k-1$ diagonals of S are within D's B-set and since the function $count\_sub\_diags$ is counting all such $k-1$-element sets in exactly this B-set, the algorithm will correctly count the set S.
Next, we show that each k-element set of diagonals is counted at most ones. To the contrary, let us assume there is a set S of k diagonals that is counted twice by the algorithm. If $k = 1$ this would mean some i_1, i_2 from the first for-loop with $i2 > i_1$ would produce the same diagonal. But this is impossible by the condition that is marked with $(*)$ in the algorithm. The latter ensures to skip those diagonals that have been counted for by a previous vertex i.
If $k > 1$, then there must be two diagonals (i_1, j_1) and (i_2, j_2) produced at the first for-loops with $i1 < i2$, that both generate S in its later course. Necessarily, (i_1, j_1) must lie in the B-set of (i_2, j_2) and vice versa. By construction this implies S is a loop. Moreover, (i_1, j_1) must

be reached from (i_2, j_2) within the search for sub-diagonals, that is, in $count\_sub\_diags(., ., ., n, i_2)$. But now the condition flagged with $(**)$ in the algorithm skips exactly this sub-diagonal, since the *root* is i_2 and the vertex i is i_1 (remember: $i1 < i_2$ by assumption). $\square$