(c) Describe an $\mathcal{O}^*(2^k)$-time algorithm which (i) takes as input a graph G and a positive integer k and (ii) outputs the set of all minimal vertex covers of G of size at most k. Prove that your algorithm is correct, and that it runs within the stated time bound. Your running time analysis must involve a recurrence relation and its solution. [10]

3. A graph $G = (V, E)$ is said to be a *split graph* if its vertex set V can be partitioned into two parts, $V = C \uplus I$, such that the set C is a clique in G and the set I is an independent set in G. For instance, every star is a split graph (but not the other way round!). Use *iterated compression* to devise an FPT algorithm for the following problem:

DELETION INTO SPLIT GRAPHS	**Parameter:** k
Input: Graph G, and $k \in \mathbb{N}$.	
Question: Does there exist a set S of at most k vertices in G such that deleting S from G results in a split graph?	

i. Describe each logical component of your iterated compression algorithm *separately* in pseudocode.

ii. Prove that your algorithm correctly solves DELETION INTO SPLIT GRAPHS.

iii. Prove an FPT upper bound on the running time of your algorithm. [20]

4. A *triangle* in a graph G is a subset of three vertices $\{x, y, z\}$ of G such that G has all the three edges $\{x, y\}, \{y, z\}, \{x, z\}$. A set of triangles in G is said to be *vertex-disjoint* if no two of them share a vertex. Note that this doesn't rule out there being *edges* between vertices of two distinct triangles in the set. In this problem we see how to check if a graph has a large set of vertex-disjoint triangles:

VERTEX-DISJOINT TRIANGLES	**Parameter:** k
Input: Graph G, $k \in \mathbb{N}$.	
Question: Does G contain a set of k vertex-disjoint triangles?	

(a) Use the *colour-coding* technique to develop a randomized FPT algorithm that solves VERTEX-DISJOINT TRIANGLES with constant probability of success. [20]

(b) Prove that your algorithm is correct, and prove that it succeeds with probability at least c for some constant $c > 0$. What is the value of c you get? [10]

(c) Prove an upper bound for the running time of your algorithm as a function of the parameter k, ignoring polynomial factors. [10]

(d) Develop a randomized FPT algorithm for this problem which has a constant probability of success and has a running time of the form $\mathcal{O}^*(2^{\mathcal{O}(k)})$; that is, the running time must be *single-exponential* in the parameter k. [20 (bonus)]