# Implement Simple H.264 Codec

**Given Source Code:**

- Download the code from here: https://vault.sfu.ca/index.php/s/VWeMsfjRP7xtEp2
- The code is written in Java and the source project is done using Eclipse with Gradle (https://gradle.org/) build system. Feel free to use the source project as is, or import the java/resource files to your own project.
- The source code skeleton includes the encoder and decoder. The empty parts of the encoder code should be implemented by the student, while the decoder code is provided as a reference for what should be expected from the encoder.
- You should fill in the "`\\\\\\\\\ FILL IN HERE/////////`" sections of the code.
- General guidelines:
    - The frame sequence is IPPP, with only the first frame being I.
    - 4:2:0 chroma subsampling is used.
    - Only 1 reference frame is used for P frames.
    - Useful reading: Overview of the H.264/AVC Video Coding Standard

**Complete the following empty parts:**

1. H26X.java:
    - Intra Prediction
        - Implement the following functions:
            - The "`Encode`" part of the `void predictIFrame(…)` function.
            - `int calculateIFrameBlockMode(...)`
            - `int[][] modePredHOR(...)`
            - `int[][] modePredVER(...)`
            - `int[][] modePredDC(...)`
        - Guidelines:

        - Use only the 16x16 Intra prediction for luma and 8x8 for chroma.
        - For each block choose between the following 3 prediction modes: Horizontal, Vertical, DC. (Refer to the intra-frame prediction section (section 4-G) of the Overview of the H.264/AVC Video Coding Standard paper.)
        - For simplicity use the same mode for each luma block and its corresponding chromas.
        - Choose the best mode based on the MAD (Mean Absolute Difference) criteria.

- Motion Vector Search
    - Implement the `SimpleEntry<Integer, Integer> logSearch(...)` function.
    - Guidelines:
        - Search for the best matching block using 2D logarithmic search.
        - Use only full sample accuracy. (No sub-pixel motion vectors).
        - Choose the best matching block based on the MAD criteria.
        - Use luma to find the motion vectors.
        - A macroblock is defined by the coordinates of its top left pixel (x,y).
        - If the best matching block is found to be (x',y'), then the motion vector would be (x'-x, y'-y).

- Inter Prediction
    - Implement the "`Encode`" part of the `void predictPFrame(...)` function.
    - Guidelines:
        - Use only the 16x16 Inter prediction.

        - Each 16x16 luma block corresponds to a 8x8 block in the chroma channels. For the corresponding chroma blocks use half the motion vector calculated for luma.

2. IntegerTransformation.java:
    - Implement the forward integer transform function (`int[][] _forwardT(…)`)
    - Guidelines:
        - Use a 4x4 integer transform for all luma and chroma components.
        - No secondary transforms (e.g. Hadamard transform for chroma or Intra_16x16 modes).
        - Useful reading: https://www.vcodex.com/h264avc-4x4-transform-and-quantization/

3. Quantizer.java:
    - Implement the quantization function (`double[][] quantize(…)`)

**To test your code:** You can preview your results using the GUI provided in the source code. Test your code using the test sequences using different QPs and note your observations.

**You should submit:**

- The Java source files after completing the missing parts. The code should compile and run without any errors. One compressed (.zip) archive. Upload to the submission system by the deadline.
- A summary page describing and commenting on your implementation details. Print the summary and bring to the class after the deadline.