```
/* format and store a lesson key into global: key */
void parseKey(char* argv[], int bpm) {
    long long int DELTA = ((DELTA_BASE * 60) / 2) / bpm;
    long long int spb = MU_S * 60 / bpm;
    for (size_t i = 0; i < KEY_SIZE; i++) {
        for (size_t j = 0; j < KEY_NUM_COLUMNS; j++)
        {
            key[i][j] = 0;
        }
    }

    for (size_t i = 0; i < KEY_SIZE; i++) {
        key[i][0] = i % 16; // button #
        key[i][1] = argv[i][0] - 48; // note type

        key[i][2] = (i + 1) * spb - DELTA + latency1;// -(60 * latency1 / bpm); //start lo
        key[i][3] = (i + 1) * spb + DELTA + latency1;// -(60 * latency1 / bpm); //start hi

        long long int offset = (spb * key[i][1]);
        key[i][4] = (i + 1) * spb - DELTA + offset + latency2;// -(60 * latency2 / bpm); // stop lo
        key[i][5] = (i + 1) * spb + DELTA + offset + latency2;// -(60 * latency2 / bpm); // stop hi
    }
}


/* format and store a lesson key into global: history */
void parseHistory(char* key[], int asize) {
    for (size_t i = 0; i < LARGE_NUMBER; i++) {
        for (size_t j = 0; j < HISTORY_NUM_COLUMNS; j++) {
            history[i][j] = 0;
        }
    }

    long long int  history_temp[LARGE_NUMBER][3];
    int history_size;

    int i = 0;
    for (size_t j = 65; j < asize; j += 3) {
        history_temp[i][0] = atoi(key, j); // button #
        history_temp[i][1] = atoi(key, j + 1); // up/down // time pressed
        history_temp[i][2] = atoi(key, j + 2); // time     //time released
        i += 1;
    }

    history_size = i;
    int k = 0;
    for (size_t i = 0; i < history_size; i++) { // take pi record and pair up key presses. assumptions: cant press same key twice without releasing inbetween
        if (history_temp[i][1] == 1) { // if down press
            for (size_t j = 0; j < history_size; j++) { // look for matching release
                if (history_temp[j][1] == 0 && // is a release
                    history_temp[j][0] == history_temp[i][0] && // same button numbers
                    history_temp[i][2] < history_temp[j][2] // start is less than stop
                    ) {
                    history[k][0] = history_temp[i][0];
                    //history[k][1] = history_temp[i][2]; // LEAVE ALONE
                    history[k][1] = history_temp[i][2] - 1184000 + 2285000 - 310000;//815000; // LEAVE ALONE
                    history[k][2] = history_temp[j][2] - 1335000 + 2420000 - 250000;//886000;
                    k += 1;
                    break;
```

```
60                  }
61              }
62          }
63      }
64  }
```