FUNCTION

/**
* The function outputs the optimal alignment with number of matches, gaps and alignment
* length
* @param[in] seq1 an array that contains RNA sequence
* @param[in] seq2 an array that contains RNA sequence
* @param[in] match_score an integer that represents match score
* @param[in] mismatch_score an integer represents mismatch score
* @param[in] gap_penalty an integer that represents gap penalty
* @param[in&out] seq1_align a char array used to store the seq1 modified alignment
* @param[in&out] seq2_align a char array used to store the seq2 modified alignment
* @param[in&out] scores a 2D array stores the modified matrix based on the algorithm
*/
void get_align(char seq1[], char seq2[], int match_score, int mismatch_score, int gap_penalty, char seq1_align[], char seq2_align[], int scores[50][50])

PRECONDITIONS

- The length of seq1[] and seq2[] must be no longer than 10
- seq1[] and seq2[] must exist
- seq1[] and seq2[] must only contains letters G, U, A, C

POSTCONDITIONS

- The optimal alignment with number of matches, gaps and alignment length must be displayed
- The size of seq1_align[] and seq2_align[] must be the same

TESTING STRETGY

Black-box testing will be used. The following test cases will be implemented:

- Function call with seq1[] and seq2[] only contains G, U, A, C, each of length smaller than or equal to 10, with integer type of match_score, mismatch_score, and gap_penalty. Compare the results generated by tests with the expected result calculated in advance. Expected output of the test: result:0, expected: 0.
- After the function call, compare the size of seq1_align[] and seq2_align[]. Expected output of the test: result:0, expected: 0.

* There is no need to test float number of match_score, mismatch_score, and gap_penalty as the float type will be converted to integer type in the main function
* The restrictions of the inputs are now moved to check_input function, and are checked by unit tests of check_input function