Student ID = up201707227
Seq1 = CTCGGA
Seq2 = CCGAAT
Match = 2
Mismatch = -1
Gap = -2

EX.1:
a)

Score Matrix:

	Gap	C	C	G	A	A	T
Gap	0	-2	-4	-6	-8	-10	-12
C	-2	2	0	-2	-4	-6	-8
T	-4	0	1	2	0	-2	-4
C	-6	-2	-1	0	4	2	0
G	-8	-4	-3	-2	2	3	4
G	-10	-6	-5	-4	0	1	5
A	-12	-8	-4	-6	-2	-1	3

Traceback Matrix:

0	3	3	3	3	3	3
2	1	1	1,3	3	3	3
2	1,2	1,2	1	3	3	3
2	2	1,2	1,2	1	1,3	3
2	2	1,2	1,2	2	1	1
2	2	1,2	1,2	2	1,2	1
2	2	1	1,2,3	2	1,2	2

1: Diagonal Move (match/mismatch)
2: Vertical Move (add gap to 1st seq)
3: Horizontal Move (add gap to 2nd seq)

b)

Best Score: 3

c)

Optimal Sequence Alignment:
CTCGGA-
C-CGAAT

d)

Only one path through the traceback matrix that resulted in the optimal score, so there is only one optimal alignment

EX.2:

a)

Score Matrix:

	Gap	C	C	G	A	A	T
Gap	0	0	0	0	0	0	0
C	0	2	2	0	0	0	0
T	0	0	1	1	0	0	2
C	0	2	2	0	0	0	0
G	0	0	1	4	2	0	0
G	0	0	0	3	3	1	0
A	0	0	0	1	5	5	3

Traceback Matrix:

0	3	3	3	3	3	3
2	1	1	1,2,3	1,2,3	1,2,3	1,2,3
2	1,2,3	1	1	1,2,3	1,2,3	1
2	1	1	1,2,3	1,2,3	1,2,3	1,2,3
2	1,2,3	1	1	3	1,2,3	1,2,3
2	1,2,3	1,2,3	1	1	1,3	1,2,3
2	1,2,3	1,2,3	2	1	1	3

1: Diagonal Move (match/mismatch)
2: Vertical Move (add gap to 1st seq)
3: Horizontal Move (add gap to 2nd seq)

b)

Best Score: 5

c)

Optimal Sequence Alignment:

CGGA	e	CGGA
CCGA		CGAA

Two paths through the traceback matrix that resulted in the optimal score, so there is two optimal alignments (multiple)

```
def main():
    # ask the sequences
    seq1 = input("Sequence 1: ")
    seq2 = input("Sequence 2: ")
    # ask the match, mismatch and gap scores
    match = int(input("Match score: "))
    mismatch = int(input("Mismatch score: "))
    gap = int(input("Gap score: "))
    # call the function to get the global alignment
    global_score, global_traceback, global_best_score, global_seq1, global_seq2, global_multiple =
global_alignment(seq1, seq2, match, mismatch, gap)
    # print the global alignment
    print("Global Alignment:")
    print("Score matrix:")
    for i in range(len(global_score)):
        print(global_score[i])
    print("Traceback matrix:")
    for i in range(len(global_traceback)):
        print(global_traceback[i])
    print("Best score:", global_best_score)
    print("Optimal sequence alignment:")
    print(global_seq1)
    print(global_seq2)
    if global_multiple:
        print("There are multiple global alignments.")
    else:
        print("There is only one global alignment.")

    print()
    print("##############################")
    print()
    # call the function to get the local alignment
    local_score, local_traceback, local_best_score, local_seq1, local_seq2, local_multiple = local_alignment(seq1,
seq2, match, mismatch, gap)
    # print the local alignment
    print("Local Alignment:")
    print("Score matrix:")
    for i in range(len(local_score)):
        print(local_score[i])
    print("Traceback matrix:")
    for i in range(len(local_traceback)):
        print(local_traceback[i])
    print("Best score:", local_best_score)
    print("Optimal sequence alignment:")
    print(local_seq1)
```

```
    print(local_seq2)
    if local_multiple:
        print("There are multiple local alignments.")
    else:
        print("There is only one local alignment.")

def global_alignment(seq1, seq2, match, mismatch, gap):
    # initialize the score matrix
    score = [[0 for i in range(len(seq2)+1)] for j in range(len(seq1)+1)]
    # initialize the traceback matrix
    traceback = [[0 for i in range(len(seq2)+1)] for j in range(len(seq1)+1)]
    # fill the first row with the gaps
    for i in range(1, len(seq1)+1):
        score[i][0] = score[i-1][0] + gap
        traceback[i][0] = 2
    # fill the first column with the gaps
    for j in range(1, len(seq2)+1):
        score[0][j] = score[0][j-1] + gap
        traceback[0][j] = 3
    # fill the rest of the matrix
    for i in range(1, len(seq1)+1):
        for j in range(1, len(seq2)+1):
            # calculate the score for the diagonal
            if seq1[i-1] == seq2[j-1]:
                score_diag = score[i-1][j-1] + match
            else:
                score_diag = score[i-1][j-1] + mismatch
            # calculate the score for the left
            score_left = score[i][j-1] + gap
            # calculate the score for the up
            score_up = score[i-1][j] + gap
            # calculate the best score
            max_score = max(score_diag, score_left, score_up)
            # fill the score matrix
            score[i][j] = max_score
            # fill the traceback matrix
            if max_score == score_diag:
                traceback[i][j] = 1
            elif max_score == score_left:
                traceback[i][j] = 3
            else:
                traceback[i][j] = 2
    # get the best score
    best_score = score[len(seq1)][len(seq2)]
    # get the optimal sequence alignment
```

```
    seq1_aligned, seq2_aligned = traceback_global(traceback, seq1, seq2)
    # check if there are multiple alignments
    multiple = False
    for i in range(1, len(seq1)+1):
        for j in range(1, len(seq2)+1):
            if score[i][j] == best_score and (traceback[i][j] == 1 or traceback[i][j] == 2 or traceback[i][j] == 3):
                multiple = True

    return score, traceback, best_score, seq1_aligned, seq2_aligned, multiple


def local_alignment(seq1, seq2, match, mismatch, gap):
    # initialize the score matrix
    score = [[0 for i in range(len(seq2)+1)] for j in range(len(seq1)+1)]
    # initialize the traceback matrix
    traceback = [[0 for i in range(len(seq2)+1)] for j in range(len(seq1)+1)]
    # fill the first row with the gaps
    for i in range(1, len(seq1)+1):
        score[i][0] = 0
        traceback[i][0] = 0
    # fill the first column with the gaps
    for j in range(1, len(seq2)+1):
        score[0][j] = 0
        traceback[0][j] = 0
    # fill the rest of the matrix
    for i in range(1, len(seq1)+1):
        for j in range(1, len(seq2)+1):
            # calculate the score for the diagonal
            if seq1[i-1] == seq2[j-1]:
                score_diag = score[i-1][j-1] + match
            else:
                score_diag = score[i-1][j-1] + mismatch
            # calculate the score for the left
            score_left = score[i][j-1] + gap
            # calculate the score for the up
            score_up = score[i-1][j] + gap
            # calculate the best score
            max_score = max(0, score_diag, score_left, score_up)
            # fill the score matrix
            score[i][j] = max_score
            # fill the traceback matrix
            if max_score == 0:
                traceback[i][j] = 0
            elif max_score == score_diag:
                traceback[i][j] = 1
```

```
            elif max_score == score_left:
                traceback[i][j] = 3
            else:
                traceback[i][j] = 2
    # get the best score
    best_score = 0
    for i in range(1, len(seq1)+1):
        for j in range(1, len(seq2)+1):
            if score[i][j] > best_score:
                best_score = score[i][j]
    # get the optimal sequence alignment
    seq1_aligned, seq2_aligned = traceback_local(traceback, seq1, seq2)
    # check if there are multiple alignments
    multiple = False
    for i in range(1, len(seq1)+1):
        for j in range(1, len(seq2)+1):
            if score[i][j] == best_score and (traceback[i][j] == 1 or traceback[i][j] == 2 or traceback[i][j] == 3):
                multiple = True
    return score, traceback, best_score, seq1_aligned, seq2_aligned, multiple


def traceback_global(traceback, seq1, seq2):
    seq1_aligned = ""
    seq2_aligned = ""
    i = len(seq1)
    j = len(seq2)
    while i > 0 and j > 0:
        if traceback[i][j] == 1:
            seq1_aligned = seq1[i-1] + seq1_aligned
            seq2_aligned = seq2[j-1] + seq2_aligned
            i -= 1
            j -= 1
        elif traceback[i][j] == 2:
            seq1_aligned = seq1[i-1] + seq1_aligned
            seq2_aligned = "-" + seq2_aligned
            i -= 1
        elif traceback[i][j] == 3:
            seq1_aligned = "-" + seq1_aligned
            seq2_aligned = seq2[j-1] + seq2_aligned
            j -= 1
    while i > 0:
        seq1_aligned = seq1[i-1] + seq1_aligned
        seq2_aligned = "-" + seq2_aligned
        i -= 1
    while j > 0:
```

```
        seq1_aligned = "-" + seq1_aligned
        seq2_aligned = seq2[j-1] + seq2_aligned
        j -= 1

    return seq1_aligned, seq2_aligned


def traceback_local(traceback, seq1, seq2):
    seq1_aligned = ""
    seq2_aligned = ""
    i = 0
    j = 0
    # find the maximum score in the matrix
    max_score = 0
    for i in range(1, len(seq1)+1):
        for j in range(1, len(seq2)+1):
            if traceback[i][j] > max_score:
                max_score = traceback[i][j]
    # find the position of the maximum score
    for i in range(1, len(seq1)+1):
        for j in range(1, len(seq2)+1):
            if traceback[i][j] == max_score:
                break
        if traceback[i][j] == max_score:
            break
    # traceback
    while traceback[i][j] != 0:
        if traceback[i][j] == 1:
            seq1_aligned = seq1[i-1] + seq1_aligned
            seq2_aligned = seq2[j-1] + seq2_aligned
            i -= 1
            j -= 1
        elif traceback[i][j] == 2:
            seq1_aligned = seq1[i-1] + seq1_aligned
            seq2_aligned = "-" + seq2_aligned
            i -= 1
        elif traceback[i][j] == 3:
            seq1_aligned = "-" + seq1_aligned
            seq2_aligned = seq2[j-1] + seq2_aligned
            j -= 1
    return seq1_aligned, seq2_aligned


main()
```

```
PS D:\faculdade\Bioinformatica\assignment3> python test.py
Sequence 1: CTCGGA
Sequence 2: CCGAAT
Match score: 2
Mismatch score: -1
Gap score: -2
Global Alignment:
Score matrix:
[0, -2, -4, -6, -8, -10, -12]
[-2, 2, 0, -2, -4, -6, -8]
[-4, 0, 1, -1, -3, -5, -4]
[-6, -2, 2, 0, -2, -4, -6]
[-8, -4, 0, 4, 2, 0, -2]
[-10, -6, -2, 2, 3, 1, -1]
[-12, -8, -4, 0, 4, 5, 3]
Traceback matrix:
[0, 3, 3, 3, 3, 3, 3]
[2, 1, 1, 3, 3, 3, 3]
[2, 2, 1, 1, 1, 1, 1]
[2, 1, 1, 1, 1, 1, 1]
[2, 2, 2, 1, 3, 3, 3]
[2, 2, 2, 1, 1, 1, 1]
[2, 2, 2, 2, 1, 1, 3]
Best score: 3
Optimal sequence alignment:
CTCGGA-
C-CGAAT
There are multiple global alignments.
```

```
Local Alignment:
Score matrix:
[0, 0, 0, 0, 0, 0, 0]
[0, 2, 2, 0, 0, 0, 0]
[0, 0, 1, 1, 0, 0, 2]
[0, 2, 2, 0, 0, 0, 0]
[0, 0, 1, 4, 2, 0, 0]
[0, 0, 0, 3, 3, 1, 0]
[0, 0, 0, 1, 5, 5, 3]
Traceback matrix:
[0, 0, 0, 0, 0, 0, 0]
[0, 1, 1, 0, 0, 0, 0]
[0, 0, 1, 1, 0, 0, 1]
[0, 1, 1, 0, 0, 0, 0]
[0, 0, 1, 1, 3, 0, 0]
[0, 0, 0, 1, 1, 1, 0]
[0, 0, 0, 2, 1, 1, 3]
Best score: 5
Optimal sequence alignment:
CG-
CGA
There are multiple local alignments.
```