GÖKSU GÜZ – 150180715

1.a)

Smith Waterman Algorithm

```
FOR 1 TO firstWord+1
    FOR 1 TO secondWord+1
        IF first words' letter is equal to second words' letter
            graph[i-1][j-1] + match
        ELSE
            graph[i-1][j-1] + mismatch;
        compare graph[i-1][j]+gap_penalty, graph[i][j-1]+gap_penalty, Si−1,j−1 + si,j , 0
        put the max to graph[i][j]
        IF max same letter is less than the compare result
            make max count letter the compare result
    ENDFOR
ENDFOR
```

Traceback

```
FOR 1 TO firstWord+1
    FOR 1 TO secondWord+1
        IF graph[i][j] is equal to the max same letter number
            WHILE graph[x][y] != 0
                lower x and y by one
            increase x and y by one
            FOR 0 TO i
                add the letter to string ptr
            FOR 0 TO size of the same words' vector
                IF the word exists in the same words' vector
                    make m = 1
            IF m is 0
                add the word to the same words' vector
    ENDFOR
ENDFOR
```

1.b) There are two for loops in my Smith Waterman algorithm code, so we have $\sum_1^n x \sum_1^n y$ = nx.ny x and y are constant values so they are cancelled and also in the traceback because there are two for loops but they are separate so the complexity is n^2 + n; n is cancelled because we take the largest number and cancel the rest (coefficients and the constants), finally the complexity is O(n^2). However, if we look at the traceback function there are nested for and while loops. We know that a while loop has O(logn) complexity which takes more time than O(n) – for loop complexity-. As we choose the longest path while we are calculating the complexity, I chose for-for-while nested loops as the longest path. The complexity of the traceback function is O(n^2logn). Thus overall complexity is O(n^2logn).

2.a) If the two words that come to the function are shorter in terms of letter number, the number of calculations will decrease, if they are longer, the more time it will take to calculate. If there are many same letter sequences between two words the number of operations we do will increase.

2.b) If the words with fewer letters come to the function, the words that will be kept in memory will decrease, if the words with less similar letter sequences come, fewer words will be recorded in the memory. If there are a lot of words that contain the same number of similar letter sequences, the space in memory will increase.

2.c) Likewise, as in the other two cases, this speed is proportional to the number of letters in the word. Here, as the number of letters decreases, the process is faster, and if less similar letter sequence including words come to the function, the operations will be faster. The important thing here is the dimensions of the graph. If the size is small, transactions will be faster. In addition, as the number of same letters in sequence increases, the program speed decreases as the traceback process takes more time.