```
class Solution {
public:
int match(char A, char B){
        return A == B;
    }
    int max_(int a, int b, int c){
        return max(max(a, b), c);
    }
    int LCS(string text1, string text2, vector<vector<int>>& grid){
        for (int i = 1; i < grid.size(); i++){
            for (int j = 1; j < grid[0].size(); j++){

                int m = match(text1[i - 1], text2[j - 1]);

                if (grid[i - 1][j] >= grid[i][j - 1] && (grid[i - 1][j] >= grid[i
- 1][j - 1] + m)){
                    grid[i][j] = grid[i - 1][j];
                }
                else if (grid[i][j - 1] >= grid[i - 1][j] && (grid[i][j - 1] >=
grid[i - 1][j - 1] + m)){
                    grid[i][j] = grid[i][j - 1];
                }
                else if ((grid[i - 1][j - 1] + m) >= grid[i - 1][j] && (grid[i -
1][j - 1] + m) >= grid[i][j - 1]){
                    grid[i][j] = grid[i - 1][j - 1] + m;
                }
            }
        }
        return grid[text1.size()][text1.size()];
    }
    string reverse(string s){
        for (int i = 0; i < s.size() / 2; i++){
            swap(s[i], s[s.size() - 1 - i]);
        }
        return s;
    }
    int longestPalindromeSubseq(string s) {
        vector<vector<int>> grid(s.size() + 1, vector<int>(s.size() + 1, {0}));

        return LCS(s, reverse(s), grid);
    }

};
```