Longest Common Subsequence using DP

Code:

```
#include <bits/stdc++.h>
using namespace std;
pair<int, string> lcs(const string &S1, const string &S2) {
 int m = S1.size();
 int n = S2.size();
 vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));
 for (int i = 1; i <= m; ++i) {
 for (int j = 1; j <= n; ++j) {
 if (S1[i - 1] == S2[j - 1])
 dp[i][j] = dp[i - 1][j - 1] + 1;
 else
 dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
 }
 }
 string lcs_str;
 int i = m, j = n;
 while (i > 0 && j > 0) {
 if (S1[i - 1] == S2[j - 1]) {
 lcs_str.push_back(S1[i - 1]);
 --i;
 --j;
 } else if (dp[i - 1][j] > dp[i][j - 1]) {
 --i;
 } else {
 --j;
 }
 }
 reverse(lcs_str.begin(), lcs_str.end());
 return {dp[m][n], lcs_str};
```

```
}
int main() {
 string S1 = "ABCBDAB";
 string S2 = "BDCAB";
 auto result = lcs(S1, S2);
 cout << "Length of LCS is " << result.first << endl;
 cout << "LCS is " << result.second << endl;
 return 0;
}
```