```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int findMinMovesMemo(int num, vector<int>& cache) {
    if (num == 1) return 0;

    if (cache[num] != -1)
        return cache[num];

    int result = 1 + findMinMovesMemo(num - 1, cache);

    if (num % 2 == 0)
        result = min(result, 1 + findMinMovesMemo(num / 2, cache));

    if (num % 3 == 0)
        result = min(result, 1 + findMinMovesMemo(num / 3, cache));

    cache[num] = result;
    return result;
}

int findMinMovesIterative(int num) {
    vector<int> table(num + 1, 0);
    table[1] = 0;

    for (int i = 2; i <= num; i++) {
        table[i] = table[i - 1] + 1;

        if (i % 2 == 0)
            table[i] = min(table[i], table[i / 2] + 1);

        if (i % 3 == 0)
            table[i] = min(table[i], table[i / 3] + 1);
    }

    return table[num];
}

int main() {
    int num;
    cout << "Enter a number: ";
    cin >> num;

    vector<int> cache(num + 1, -1);
    int memoResult = findMinMovesMemo(num, cache);
    cout << "\n[Memoization] Minimum moves to make " << num << " become 1: " <<
memoResult << endl;
```

```
    int iterResult = findMinMovesIterative(num);
    cout << "[Iterative DP] Minimum moves to make " << num << " become 1: " << iterResult <<
endl;

    cout << "\nNote: Both methods give the same answer, but the iterative one is faster for
big numbers.\n";
    return 0;
}
```