Rod Cutting Problem #DynamicProgramming

```cpp
/*
 * Rod Cutting Problem : Dynamic Programming
 * Given a rod of length n and an array of prices that contains prices of all pieces of size smaller than
    length   | 1   2   3   4   5   6   7   8
    ---------------------------------------
    price    | 1   5   8   9  10  17  17  20

    And if the prices are as following, then the maximum obtainable value is 24 (by cutting in eight piece

    length   | 1   2   3   4   5   6   7   8
    ---------------------------------------
    price    | 3   5   8   9  10  17  17  20
 */
int price[9] = {0, 1, 5, 8, 9, 10, 17, 17, 20};
bool done[9];
int value[9];
void init() {
    for (int i = 0; i < 9; ++i) done[i] = false, value[i] = 0;
}
int cutRodRecursion1(int n) {
    if (n <= 0) return 0;
    int maxval = INT_MIN;
    for (int i = 1; i <= n; ++i) {
        int temp;
        if (done[n-i]) temp = value[n-i];
        else temp = cutRodRecursion1(n - i), done[n-i] = true, value[n-i] = temp;
        maxval = max(maxval, price[i] + temp);
    }
    value[n] = maxval;
    return value[n];
}
int cutRodRecursion2(int n) {
    if (done[n]) return value[n];
    done[n] = true;
    if (n <= 0) return 0;
    int maxval = INT_MIN;
    for (int i = 1; i <= n; ++i) {
        maxval = max(maxval, price[i] + cutRodRecursion2(n - i));
    }
    value[n] = maxval;
    return value[n];
}
int cutRodIterative(int n) {
    int val[n+1];
    val[0] = 0;
    for (int i = 1; i <= n; ++i) {
        int maxval = INT_MIN;
        for (int j = 1; j <= i; ++j) {
            maxval = max (maxval, price[j] + val[i-j]);
        }
        val[i] = maxval;
    }
    return val[n];
}
int main() {
//    freopen("input", "r", stdin);
    init();
    cout << cutRodRecursion1(8) << endl;
    init();
    cout << cutRodRecursion2(8) << endl;
```

```
    cout << cutRodIterative(8) << endl;

    return 0;
}
```