The Power Sum #Backtracking

See the input output given below if you've forgotten what the problem was all about.

- Problem Link
- Editorial

```cpp
int ipow (int b, int e) { return (e == 0) ? 1 : b * ipow(b, e-1); }

ll sol (ll n, ll p, ll sum, ll last) {
    if (sum == n) {
        return 1;
    }
    ll total = 0;
    ++last;
    while (sum + ipow(last, p) <= n) {
        total += sol(n, p, sum + ipow(last, p), last);
        ++last;
    }
    return total;
}

int count_expression (int x, int n, vector<int>& vals) {
    int s = 0;
    for (auto v : vals) {
        s += ipow(v, n);
    }
    if (s == x) return 1;
    else {
        int answer = 0;
        int v = vals.empty() ? 1 : vals.back() + 1;
        while (s + ipow(v, n) <= x) {
            vals.push_back(v);
            answer += count_expression(x, n, vals);
            vals.pop_back();
            ++v;
        }
        return answer;
    }
}

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);

    int x, n;
    cin >> x >> n;
    vector<int> vals;
    cout << count_expression(x, n, vals) << endl;
    // cout << sol(x, n, 0, 0) << endl;
```

```
    return 0;
}
```

In the above solution both the function `sol()` and `count_expression()` works perfectly. Though `sol()` doesn't take a vector, since we don't need to print how the sum can be formed.

input:

> 100 2

output:

> 3

Because: $100 = (10_2) = (6_2 + 8_2) = (1_2 + 3_2 + 4_2 + 5_2 + 7_2)$