数据结构作业（3）

问题一

Solution Class

```
class Solution {
public:
    double solve(double x, int n, double & res) {
    if (n == 1) {
            res = x;
            return x;
        }
        double lst = solve(x, n-1, res);
        double cur = -lst * x * x / (2*n - 1) / (2*n - 2);
        res += cur;
        return cur;
    }
};
```

Main Function

```
int main() {
    Solution s;
    double res = 0;
    double x; int n; cin >> x >> n;
    s.solve(x, n, res);
    cout << res << endl;

    return 0;
}
```

问题二

使用递归

```
class Solution {

    int total;
    vector<vector<int>> res;

public:
    Solution(int total) : total(total) {}

    // driver function
    vector<vector<int>> upstairs() {
        vector<int> path;
        int rest = total;
        path.push_back(1); rest -= 1;
        upstairs(rest, path);
        path.pop_back(); rest += 1;
        path.push_back(2); rest -= 2;
        upstairs(rest, path);
        return res;
    }

    void upstairs(int rest, vector<int> path) {
        if (rest < 0) return;
        else if (rest == 0) res.push_back(path);
        else {
            path.push_back(1); rest -= 1;
            upstairs(rest, path);
            path.pop_back(); rest += 1;
            path.push_back(2); rest -= 2;
            upstairs(rest, path);
        }
    }
};
```

动态规划（未优化）

```
class Solution {
    int total;
    vector<vector<int>> res;
public:
    vector<vector<int>> upstairs() {
        vector<vector<vector<int>>> dp(total+1);
        dp[1].push_back({1});
        dp[2].push_back({1, 1}); dp[2].push_back({2});
        for (int i = 3; i <= total; ++i) {
            for (size_t j = 0; j < dp[i-1].size(); ++j) {
                vector<int> tmp = dp[i-1][j];
                tmp.push_back(1);
                dp[i].push_back(tmp);
            }
            for (size_t j = 0; j < dp[i-2].size(); ++j) {
                vector<int> tmp = dp[i-2][j];
                tmp.push_back(2);
                dp[i].push_back(tmp);
            }
        }
        for (auto & v : dp[total]) {
            res.push_back(v);
        }
    }
}
```

问题三

```
#include <iostream>
#include <string>
#include <cmath>
#include <stack>
#include <regex>
#include <cassert>

using namespace std;


double calculate(string expr) {
    stack<double> nums;
    stack<char> ops;
    int len = expr.size();
    enum PRIORTY {ADD = 1, SUB = 1, MUL = 2, DIV = 2, POW = 3};
    for (int i = 0; i < len; ++i) {
        if (expr[i] == ' ') continue;
        if (isdigit(expr[i])) {
            int j = i;
            while (j < len && isdigit(expr[j])) ++j;
            nums.push(stod(expr.substr(i, j-i)));
            i = j-1;
        } else if (expr[i] == '(') {
            ops.push(expr[i]);
        } else if (expr[i] == ')') {
            while (ops.top() != '(') {
                double num2 = nums.top(); nums.pop();
                double num1 = nums.top(); nums.pop();
                char op = ops.top(); ops.pop();
                double res;
                switch (op) {
                    case '+': res = num1 + num2; break;
                    case '-': res = num1 - num2; break;
                    case '*': res = num1 * num2; break;
                    case '/': res = num1 / num2; break;
                    case '^': res = pow(num1, num2); break;
                }
                nums.push(res);
            }
            ops.pop();
        } else {
            while (!ops.empty() && ops.top() != '(' && PRIORTY(expr[i]) <= PRIORTY(ops.top())) {
                double num2 = nums.top(); nums.pop();
                double num1 = nums.top(); nums.pop();
                char op = ops.top(); ops.pop();
                double res;
                switch (op) {
```

```
                    case '+': res = num1 + num2; break;
                    case '-': res = num1 - num2; break;
                    case '*': res = num1 * num2; break;
                    case '/': res = num1 / num2; break;
                    case '^': res = pow(num1, num2); break;
                }
                nums.push(res);
            }
            ops.push(expr[i]);
        }
    }
    while (!ops.empty()) {
        double num2 = nums.top(); nums.pop();
        double num1 = nums.top(); nums.pop();
        char op = ops.top(); ops.pop();
        double res;
        switch (op) {
            case '+': res = num1 + num2; break;
            case '-': res = num1 - num2; break;
            case '*': res = num1 * num2; break;
            case '/': res = num1 / num2; break;
            case '^': res = pow(num1, num2); break;
        }
        nums.push(res);
    }
    assert(nums.size() && ops.size() == 0);
    return nums.top();
}


int main() {

    // for integer expression

    /*example
    > input the expression:
    > ((x+2)^2 - 4) / 4
    > input the value of variable:
    > x = 2

    then the expression will be changed to:
    > ((2+2)^2 - 4) / 4
    then the calculator will calculate the expression and print the result:
    > the result is: 3
    */

    cout << "input the expression: " << endl;
    string expr; getline(cin, expr);
    cout << "input the value of variable:\n(if there are more than one variable, please use ','
    string vars; getline(cin, vars);
    // x = 2, y=3 z = 4
```

```
    regex reg("[a-zA-Z] *= *[0-9]+"); // search the variable and its value
    smatch sm;
    stack<string> st;
    while (regex_search(vars, sm, reg)) {
        st.push(sm[0]);
        vars = sm.suffix();
    }
    while (!st.empty()) {
        string tmp = st.top(); st.pop();
        regex reg2("[a-zA-Z]"); // search the variable
        smatch sm2;
        regex_search(tmp, sm2, reg2);
        string var = sm2[0];

        regex reg3("[0-9]+"); // search the value
        smatch sm3;
        regex_search(tmp, sm3, reg3);
        string val = sm3[0];

        regex reg4(var);
        expr = regex_replace(expr, reg4, val);
    }
    // cout << "the expression will be changed to: " << endl;
    // cout << expr << endl;

    cout << "the result is: " << endl;
    cout << calculate(expr) << endl;

    return 0;
}
```

问题四

```cpp
using ll = long long;

class Solution {

    int n, m;

public:
    Solution(int n, int m) : n(n), m(m) {}

    ll Ackerman(int n, int m) {
        if (n == 0) return 1;
        if (m == 0) {
            if (n == 1) return 2;
            else return n + 2;
        }
        return Ackerman(Ackerman(n - 1, m), m - 1);
    }

    ll Ackerman_non_recursive(int n, int m) {
        stack<vector<ll>> s; // simulate the recursive stack
        s.push({n, m, 0});
        stack<ll> res; // simulate the recursive return value

        while (!s.empty()) {
            auto v = s.top(); s.pop();
            int n_cur = v[0], m_cur = v[1], state = v[2];

            // the condition of the end of the loop
            if (n_cur == 0) {
                res.push(1);
                continue;
            } else if (m_cur == 0) {
                if (n_cur == 1) res.push(2);
                else res.push(n_cur + 2);
                continue;
            }

            // change the recursive operation to stack operation
            if (state == 0) {
                s.push({n_cur, m_cur, 1});
                s.push({n_cur - 1, m_cur, 0});
            } else if (state == 1) {
                s.push({res.top(), m_cur - 1, 0});
                res.pop();
            }
        }
```

```
        return res.top();
    }

};
```