```cpp
#include <iostream>
#include <iomanip>
#include <vector>
#include <chrono>
#include "ABS.h"
#include "ABQ.h"
using namespace std;

int main() {
    vector<double> scaleVec {1.5, 2.0, 3.0, 10.0, 50.0,
100.0};                          // 50 added due to 10-100 being a large jump
    //vector<int> nVec {10'000, 30'000, 50'000, 75'000, 100'000,
150'000};               // N values for unoptimized(slow) dequeue
    vector<int> nVec {10'000'000, 30'000'000, 50'000'000, 75'000'000,
100'000'000};     // N values now that dequeue isn't terrible

    std::chrono::time_point<std::chrono::system_clock> start, end;
    std::chrono::duration<double> duration;
    ABS<int> *stack;
    ABQ<int> *queue;
    int resizes;

    cout << left;

    for (const auto& scale: scaleVec) {
        for (const auto& n: nVec) {
            // Test stack push
            stack = new ABS<int>(2, scale);
            start = std::chrono::system_clock::now();
            for (int i{}; i < n; ++i) {
                stack->push(i);
            }

            end = std::chrono::system_clock::now();
            duration = end - start;
            resizes = stack->getTotalResizes();
            cout << setw(14) << "Stack-Push" << " SF: " << setw(3) << scale << "
N: " << setw(9) << n << " Resizes: " << setw(2) << resizes << " Duration: " <<
duration << endl;

            // Test stack pop
            start = std::chrono::system_clock::now();
            for (int i{}; i < n; ++i) {
                stack->pop();
            }
            end = std::chrono::system_clock::now();
            duration = end - start;
            cout << setw(14) << "Stack-Pop" << " SF: " << setw(3) << scale << "
N: " << setw(9) << n << " Resizes: " << setw(2) << stack->getTotalResizes() -
resizes << " Duration: " << duration << endl;
            delete stack;

            // Test queue enqueue
            queue = new ABQ<int>(2, scale);
            start = std::chrono::system_clock::now();
            for (int i{}; i < n; ++i) {
                queue->enqueue(i);
            }
            end = std::chrono::system_clock::now();
            duration = end - start;
            resizes = queue->getTotalResizes();
            cout << setw(14) << "Queue-Enqueue" << " SF: " << setw(3) << scale <<
" N: " << setw(9) << n << " Resizes: " << setw(2) << queue->getTotalResizes() <<
" Duration: " << duration << endl;

            // Test queue dequeue
            start = std::chrono::system_clock::now();
            for (int i{}; i < n; ++i) {
                queue->dequeue();
            }
            end = std::chrono::system_clock::now();
            duration = end - start;
            cout << setw(14) << "Queue-Dequeue" << " SF: " << setw(3) << scale <<
" N: " << setw(9) << n << " Resizes: " << setw(2) << queue->getTotalResizes() -
resizes << " Duration: " << duration << endl;
            delete queue;
        }
    }
}
```