```
=== 演習1 解答例 ============================
--- compile-and-run.txt ---
$ g++ -std=c++17 ex05-1.cpp
$ echo 1 2 3 4 5 | ./a.out
5 4 3 2 1


--- ex05-1.cpp ---
// 入力の逆順に出力
#include <iostream>
#include <stack>

int main()
{
   std::stack<int> s;
   for (int x; std::cin >> x; )
      s.push(x);

   while (!s.empty()) {
      std::cout << s.top() <<" ";
      s.pop();
   }
   std::cout <<"\n";
}
```

```
=== 演習2 解答例 ============================
--- compile-and-run.txt ---
$ g++ -std=c++17 ex05-2.cpp
$ ./a.out 10
3
$ ./a.out 100
4.15
$ ./a.out 200
3.83


--- random.hpp ---
// 乱数生成
// https://ja.wikipedia.org/wiki/メルセンヌ・ツイスタ
#include <random>
class UniDist {
   std::random_device seed;
   std::mt19937 engine;
   //std::default_random_engine engine{}; // for debug
   std::uniform_int_distribution<int> udist;
public:
   UniDist(int first, int last)
      :seed{}, engine{seed()}, udist{first,last}{}
   // [first, last] の一様乱数
   auto get(){ return udist(engine); }
};

class ExpDist{ // unit時間にtimes回発生する条件の乱数
   std::random_device seed;
   std::mt19937 engine;
   std::exponential_distribution<double> edist;
   double unit;
public:
   ExpDist(double lambda, double u =1.0)
      :seed{}, engine{seed()}, edist{lambda}, unit{u}{}
   // 次に起こるまでの時間
   auto get(){ return edist(engine)*unit; }
};


--- ex05-2.cpp ---
#include <iostream>
#include <queue>
#include "random.hpp"

// 引数の数の患者の診察が終了した時点の待ち人数
int simulate(int num_patient =100) {
   ExpDist next_patient {6.0}; // 次の患者到着時間
   ExpDist clinical_time{7.5}; // 診察時間
   double arrival{ next_patient.get() };
   std::queue<double> q; // 先頭は診察中, 他が待ち
   while (num_patient > 0) {
      if (q.empty() || arrival < q.front()) {
         if (!q.empty()) q.front() -= arrival;
         q.push( clinical_time.get() ); // 到着
         arrival = next_patient.get();
      } else {
         arrival -= q.front();
         q.pop(); // 診察終了
         -- num_patient;
      }
   }
   return q.size();
}

int main(int argc, char* argv[])
{
   const int N{ argc>1 ? std::atoi(argv[1]):10 };
   double sum{};
   for (int i = 0; i < N; i++) {
```

```
        auto x {simulate()};
        // std::cout << x <<" ";
        sum += x;
    }
    std::cout <<sum/N<<"\n";
}
```

```
=== 演習3 解答例 ============================
--- compile-and-run.txt ---
g++ -std=c++17 ex05-3.cpp
$ echo '[()]{}{[()()]()}' | ./a.out
ok
$ echo '[(])' | ./a.out
unmatch ]


--- ex05-3.cpp ---
// カッコの対応づけ
//  true for [()]{}{[()()]()} and false for [(]).
#include <iostream>
#include <stack>

bool match(std::stack<char>& stk, char ch)
{
   char tch { (ch==']') ? '[':
              (ch=='}') ? '{':
              (ch==')') ? '(':'\0' };
   if (stk.top() != tch)
      return false;
   stk.pop();
   return true;
}

int main()
{
   std::stack<char> stk;
   bool flag{true};
   char ch;
   while (flag && std::cin >> std::noskipws >> ch) {
      switch (ch) {
      case '[': case '{': case '(':
         stk.push(ch);
         break;
      case ']': case '}': case ')':
         flag = match(stk, ch);
         break;
      }
   }
   if (flag) {
      std::cout << "ok\n";
   } else {
      std::cout << "unmatch "<<ch<<"\n";
   }
}
```

```
=== 演習4 解答例 ============================
--- compile-and-run.txt ---
$ g++ -std=c++17 ex05-4.cpp
$ ./a.out
たけやぶやけた
しんぶんし


--- ex05-4.cpp ---
// スタックとキューで回文判定
#include <iostream>
#include <vector>
#include <stack>
#include <queue>
using std::cout, std::string, std::vector;

bool is_palindrome(const vector<string>& stmt) {
   std::queue<string> q;
   std::stack<string> s;
   for (auto e : stmt) {
      q.push(e);
      s.push(e);
   }

   while (!q.empty()) {
      if (q.front() != s.top())
         return false;
      q.pop();
      s.pop();
   }
   return true;

}

int main()
{
   vector<vector<string>> v {
   {"た","け","や","ぶ","や","け","た"},
   {"し","ん","か","ん","せ","ん"},
   {"し","ん","ぶ","ん","し"} };

   for (const auto& s:v) {
      if (is_palindrome(s)) {
         for (auto& e:s)
            cout << e;
         cout <<"\n";
      }
   }
}
```

```
=== 演習5 解答例 ============================
--- compile-and-run.txt ---
$ g++ -std=c++17 ex05-5.cpp
$ ./a.out
13 17 + 4 1 - *
90
$ echo '3 4 + 1 2 - *' | ./a.out
-7


--- ex05-5.cpp ---
// Reverse Polish Notation
// https://ja.wikipedia.org/wiki/逆ポーランド記法
#include <iostream>
#include <stack>
#include <cctype>

int main()
{
   std::stack<int> stk;
   for (std::string token; std::cin >> token; ) {
      if (std::isdigit(token[0]))
         stk.push( std::stoi(token) );
      else {
         if (stk.size() < 2) {
            std::cout <<"error\n";
            break;
         }
         int x { stk.top() }; stk.pop();
         int y { stk.top() }; stk.pop();
         if (token == "+")
            stk.push(y + x);
         else if (token == "-")
            stk.push(y - x);
         else if (token == "*")
            stk.push(y * x);
         else if (token == "/" && x != 0)
            stk.push(y / x);
         else {
            std::cout <<"error\n";
            break;
         }
      }
   }
   if (stk.size() == 1) {
      std::cout << stk.top() <<"\n";
      stk.pop();
   } else {
      std::cout <<"error\n";
   }
}
```