```
#include <iostream>
#include <random>
#include <string>
#include <map>
#include <tuple>
#include <fstream>
using namespace std;

void save_to_file(char query, const auto& results, const auto& other_data) {
  if (query == 'n') return;
  else if (query != 'y') return;

  cout << "Would you like to save the hits per number in a seperate file from
total/average/sides/rolls? y/n\n";
  char seperate_files;
  cin >> seperate_files;

  if (seperate_files == 'y') {
        cout << "Hits file name:\n";

        string hits_filename;
        cin >> hits_filename;

        ofstream hits_file;
        hits_file.open(hits_filename + ".txt");

        for (auto [key, val] : results) {
        hits_file << key << ":\t" << val << '\n';
        }

        hits_file.close();


        cout << "Other data file name:\n";

        string other_filename;
        cin >> other_filename;

        ofstream other_file;
        hits_file.open(other_filename + ".txt");

        hits_file << "Total:\t" << get<0>(other_data)
        << "\nAverage:\t" << get<1>(other_data)
        << "\nSides:\t" << get<2>(other_data)
```

```
            << "\nRolls:\t" << get<3>(other_data) << '\n';

            hits_file.close();
  } else {
            cout << "File name:\n";
            string filename;
            cin >> filename;

            ofstream output;
            output.open(filename + ".txt");

            for (auto [key, val] : results) {
            output << key << ":\t" << val << '\n';
            }

            output << "\nTotal:\t" << get<0>(other_data)
            << "\nAverage:\t" << get<1>(other_data)
            << "\nSides:\t" << get<2>(other_data)
            << "\nRolls:\t" << get<3>(other_data) << '\n';

            output.close();
  }
  return;
}

int get_positive_input() {
  string response;
  cin >> response;

  try {
            int response_number = stoi(response);
            if (response_number > 0) return response_number;
  } catch (invalid_argument&) {
            return -1;
  }

  return -1;
}

int main() {
  while (true) {
            cout << "\n==Dice Roller==\n" << "\n^_^ How many sides do you want your dice to
have? ^_^\n";
            const auto dice_sides = get_positive_input();
```

```
        if (dice_sides == -1) {
        cout << "\nNot a valid input, try again.\n";
        continue;
        }

        // cout << dice_sides << '\n';

        cout << "\n^_^ How many times would you like to roll the dice? ^_^\n";
        const auto rolls = get_positive_input();

        if (rolls == -1) {
        cout << "\nNot a valid input, try again.\n";
        continue;
        }

        // cout << rolls << '\n';

        const auto [other_data, results] = [rolls, dice_sides]() -> pair<tuple<int, double, int, int>,
map<int, int>> {
        map<int, int> results;
        random_device rd_seed;
        uniform_int_distribution<int> dist(1, dice_sides);

        for (int i = 0; i < rolls; ++i) {
        ++results[dist(rd_seed)];
        }

        int sum = 0;
        for (auto [key, val] : results) {
        sum += key * val;
        }

        return { {sum, (double) sum / (double) rolls, dice_sides, rolls}, results};
        }();


        cout << "\n\n--+ RESULTS +--\n\n";
        for (auto [key, val] : results) {
        cout << key << ":\t" << val << '\n';
        }

        cout << "\nTotal:\t" << get<0>(other_data)
        << "\nAverage:\t" << get<1>(other_data)
```

```
          << "\nSides:\t" << get<2>(other_data)
          << "\nRolls:\t" << get<3>(other_data) << '\n';

          cout << "\n\nWould you like to save your data? y/n\n";
          char query;
          cin >> query;

          save_to_file(query, results, other_data);
  }
}
```