Computer script used in statistical analysis

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <tr1/unordered_set>


int main()
{
    std::string sequence_file("input.txt");
    std::ifstream input(sequence_file.c_str());

// find number of sequences
    std::size_t num_sequence = 0;
    std::string ts;
    while(input.peek() != EOF)
    {
        num_sequence++;
        getline(input, ts);
        getline(input, ts);
        getline(input, ts);
    }

// read in sequences
    input.clear();
    input.seekg(0, std::ios::beg);

    std::vector<std::string> sequence(num_sequence);
    for(std::size_t i = 0; i < num_sequence; ++i)
    {
        getline(input, ts);
        getline(input, sequence[i]);
        getline(input, ts);
    }
```

```
for(std::size_t epitope_length = 6; epitope_length <= 30; ++epitope_length)
{
    // find unique sub_strings
    std::tr1::unordered_set<std::string> unique_epitopes;

    for(std::size_t i = 0; i < num_sequence; ++i)
        for(std::size_t j = 0; j < sequence[i].size() - epitope_length + 1; ++j)
            unique_epitopes.insert(sequence[i].substr(j, epitope_length));

    std::vector<std::string> epitopes(unique_epitopes.size());
    std::uninitialized_copy(unique_epitopes.begin(),
                            unique_epitopes.end(),
                            epitopes.begin());

    // count occurances of each epitope and output results
    std::cout << "Epitope Length: " << epitope_length << std::endl;
    std::cout << "Number of Unique Epitopes: " << epitopes.size() << std::endl;

    for(std::size_t i = 0; i < epitopes.size(); ++i)
    {
        std::cout << epitopes[i] << "\t";

        std::vector<std::size_t> in_sequence(num_sequence, 0);
        std::size_t count = 0;

        for(std::size_t j = 0; j < num_sequence; ++j)
        {
            if(sequence[j].find(epitopes[i]) != std::string::npos)
            {
                count++;

                std::size_t p = 0;
                while(1)
                {
                    p = sequence[j].find(epitopes[i], p);
                    if(p == std::string::npos)
                        break;
                    in_sequence[j]++;
                    p++;
                }
            }
        }

        std::cout << count << "\t";

        std::cout << in_sequence[0];
        for(std::size_t j = 1; j < num_sequence; ++j)
            std::cout << "," << in_sequence[j];

        std::cout << std::endl;
    }
    std::cout << std::endl;
}

}
```