EIT, 306387, Gabriel Ćwiek, MAPT 2, lab 4, https://github.com/Astaree/MAPT_2

1. Ex.1:

```cpp
#include <iostream>
#include <vector>
using namespace std;

template <typename T>
void display_vec(const vector<T>& c){
  for (T i : c) {
    cout << i << " ";
  }
  cout << endl;
}

int main() {
  vector<int> v;
  for (size_t i=0; i<10; ++i){
    v.push_back(i);
  }
  display_vec(v);
  cout<<v.size()<<endl;
  v.erase(v.begin()+3, v.begin()+7);
  display_vec(v);
  v.pop_back();
  display_vec(v);
  v.insert(v.begin(),102);
  display_vec(v);
  v.push_back(110011);
  display_vec(v);
}
```

2. Ex. 2:

```
#include <iostream>
#include <queue>

using namespace std;

//fill queue with consecutive integers
template <typename T>
void fillQ(queue<T>& q, int n){
    for (int i = 1; i <= n; i++) {
        q.push(i);
    }
}

 //output the queue
template <typename T>
void displayQ(const queue<T>& q){
    queue<T> q2 = q;
    while (!q2.empty()) {
        cout << q2.front() << " ";
        q2.pop();
    }
    cout << endl;
}
//chanfe the size of the queue and if the new size is smaller than the old
one, remove the elements from the front and
//cout "removed"
//if the new size is larger than the old one, do nothing
template <typename T>
void changeQsize(queue<T>& q, int n){
    if (n < q.size()) {
        for (int i = 0; i < q.size() - n; i++) {
            q.pop();
            cout << "removed" << endl;
        }
    }
}


//Using the function, add 4 elements to the queue
template <typename T>
void add4(queue<T>& q){
    for (int i = 0; i < 4; i++) {
        q.push(i);
    }
}
```

```
int main() {
    queue<int> q;
    // Inserting integer numbers from 1 to 12 into the queue
    fillQ(q, 12);
    displayQ(q);
    // Change the size of the queue to 5
    changeQsize(q, 5);
    displayQ(q);
    // Change the size of the queue to 10
    changeQsize(q, 10);
    displayQ(q);
    // Add 4 elements to the queue
    add4(q);
    displayQ(q);
}
```

3. Ex. 3:

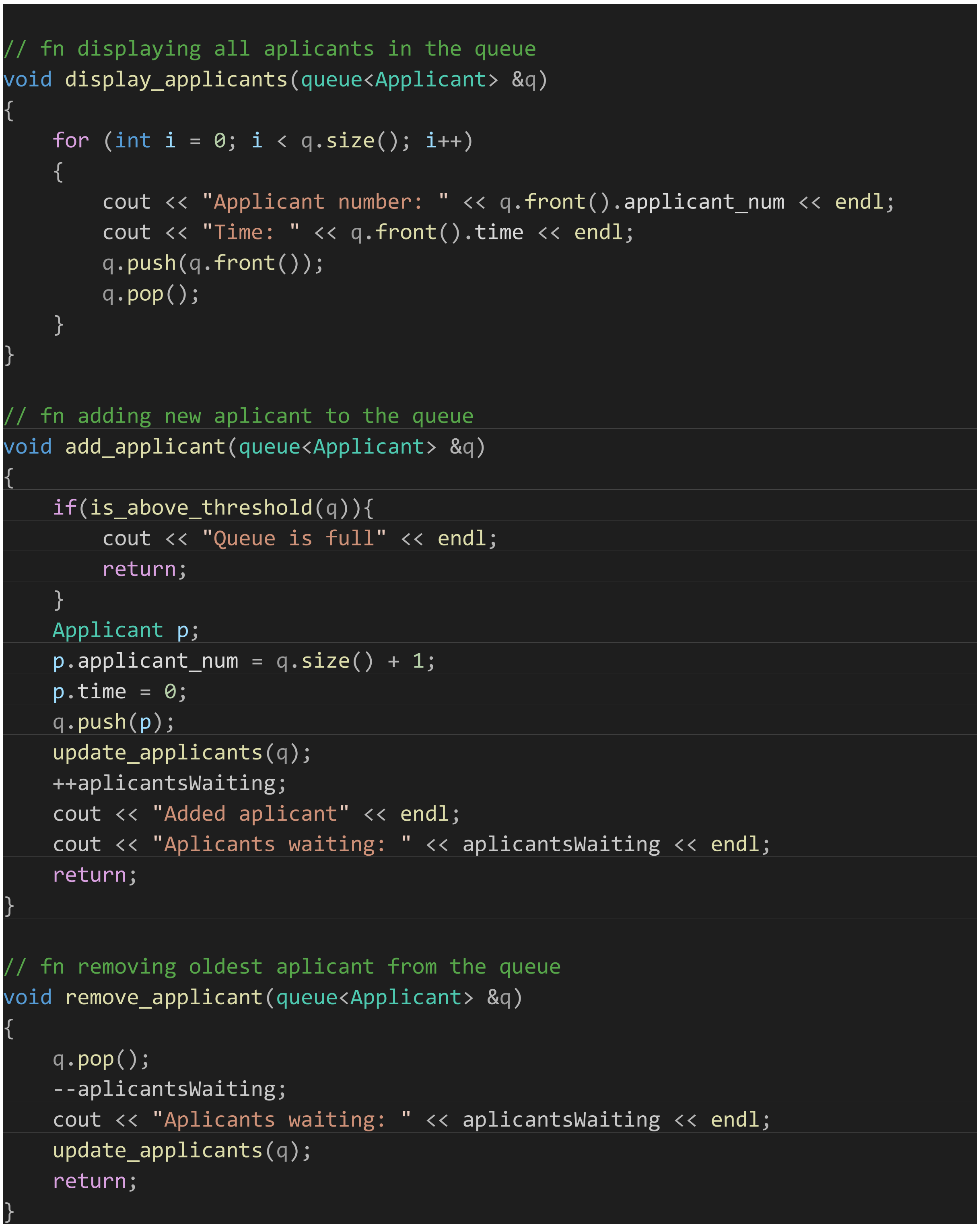
```
#include <iostream>
#include <queue>
using namespace std;

int aplicantsWaiting = 0; // im lazy
int threshold = 5; // im still lazy

class Applicant
{
public:
    int applicant_num = 0;
    int time;
};

bool is_above_threshold(queue<Applicant> &q);
void display_applicants(queue<Applicant> &q);
void add_applicant(queue<Applicant> &q);
void remove_applicant(queue<Applicant> &q);
void update_applicants(queue<Applicant> &q);

// fn testing if the queue is above threshold, if so return true
bool is_above_threshold(queue<Applicant> &q)
{
    if (q.size() > threshold)
    {
        return true;
    }
    return false;
}
```



```cpp
// fn displaying all aplicants in the queue
void display_applicants(queue<Applicant> &q)
{
    for (int i = 0; i < q.size(); i++)
    {
        cout << "Applicant number: " << q.front().applicant_num << endl;
        cout << "Time: " << q.front().time << endl;
        q.push(q.front());
        q.pop();
    }
}

// fn adding new aplicant to the queue
void add_applicant(queue<Applicant> &q)
{
    if(is_above_threshold(q)){
        cout << "Queue is full" << endl;
        return;
    }
    Applicant p;
    p.applicant_num = q.size() + 1;
    p.time = 0;
    q.push(p);
    update_applicants(q);
    ++aplicantsWaiting;
    cout << "Added aplicant" << endl;
    cout << "Aplicants waiting: " << aplicantsWaiting << endl;
    return;
}

// fn removing oldest aplicant from the queue
void remove_applicant(queue<Applicant> &q)
{
    q.pop();
    --aplicantsWaiting;
    cout << "Aplicants waiting: " << aplicantsWaiting << endl;
    update_applicants(q);
    return;
}
```

```
// fn updating all aplicants time and number
void update_applicants(queue<Applicant> &q)
{
    for (int i = 0; i < q.size(); i++)
    {
        q.front().time++;
        q.front().applicant_num = i + 1;
        q.push(q.front());
        q.pop();
    }
}

int main()
{
    queue<Applicant> eit;

    // adding 5 applicants to the queue
    for (int i = 0; i < 5; i++)
    {
        add_applicant(eit);
    }

    // displaying all applicants
    display_applicants(eit);

    //last applicant number
    cout << "Last applicant number: " << eit.back().applicant_num << endl;

    remove_applicant(eit);
    remove_applicant(eit);
    display_applicants(eit);
    //waiting for service
    cout << "Waiting for service: " << aplicantsWaiting << endl;

}
```