Task 5 (8 points)

The goal of the task is to simulate the work of a bank window, which serves clients. Each client is represented as a `Client` class object. Clients go to the bank window represented by the `BankWindow` class. The key element of each window is the queue (`Queue` class) in which customers set up. The queue has a certain capacity and when it is full, clients are sent back.

Do not modify main.cpp except uncommenting parts of the task.

Part 1 (1p.)

In the `Client` class, implement all methods and functions. The class has two members: `arrival_time` and `service_time`. The first is the client's arrival time (in a specific minute of simulation), while the second is the time necessary to serve the client (in minutes). The class has an argumentless (default) constructor that should initialize both class members to zero.

The `void set(long arrival_time)` method sets the corresponding `arrival_time` field value, while `service_time` member should be set to 10 (we initially assume a fixed customer service time of 10 minutes). Methods int `int what_time() const` and `int how_long() const` return the values of `arrival_time` and `service_time` respectively. Overloaded operator<< will be required to print information about the client (match formatting to the example output).

Part 2 (5p.)

The `Queue` class represents a queue data structure realized on a one-directional linked list. In the private part, the class has a nested structure of the list node (`Node`), which consists fields: `item` (of type `Client`, representing single element in the queue) and `next` (type `struct Node*`, which is a pointer to the next node).

In the `Queue` class, we also have a `front` which is a pointer to the beginning of the list (queue), a `rear` which is a pointer to the end of the list (queue), `clients` (current number of clients in the queue) and `qsize` (a queue capacity). The `Queue` constructor creates an empty queue with a given capacity (or default Q_SIZE).

The queue is a FIFO type structure, therefore the operations of adding and removing elements are limited to adding at the end of the queue (`enqueue` () method) and removing from the beginning of the queue (`dequeue` () method). The `bool enqueue(const Client &client)` method should allocate the new node, set its components accordingly, and then perform the operation of adding the node at the end of the queue. Please remember to handle different cases and increase the `clients` count. A client cannot be added to the queue, if the queue is already full. The `bool dequeue(Client &client)` method should get (remove) a client from the front of the queue. The client taken from the queue should be placed in the `client` variable (passed as a method argument). Please remember to handle different cases and reduce the `clients` counter. The client cannot be retrieved from the queue if the queue is already empty. Both methods return a logical value indicating the success or failure of the operation.

The `bool isempty() const`, `bool isfull() const` and `int size() const` methods return information whether the queue is empty, full and the current number of clients in the queue. You must implement all necessary methods and functions. Do not forget about the correct destructor. The `ostream&`

`print(ostream&) const` method is used to print information about the content of the queue (match formatting to the example output).

Part 3 (1p.)

The main element of the `BankWindow` class are: `queue_to_window` - the queue of customers to be served and `working_time` - the window's working time (in hours). These fields must be handled in the constructor.

Other member fields of this class are initialized (set to zero) in the class declaration and are used at a later stage of class usage (during simulation). The `release_time` member is the time during which the window is occupied (unavailable) and refers to the time needed to settle the client's case (`service_time` member in `Client` class).

The other members in the class are the necessary to calculate the simulation statistics:

- `number_returned` - the number of customers sent back (because the queue was full);
- `number_accepted` - number of all clients accepted into the queue;
- `number_served` - the number of clients served (taken from the queue to the window);

You will need the operator << to print information about the status of the window (match formatting to the example output).

Part 4 (1 p.)

The `void simulation(int time_between_clients)` method is used to simulate the operation of a bank window. The simulation is carried out in one minute cycles. In each cycle you should:

- Check if a new client appears. The new client appears exactly every `time_between_clients` minutes (you can control process by checking cycle%time_between_clients).
- If new client appeared, then we add them to the queue, if the queue is not yet full. In this case we sent back the client. Remember to control the value of `number_returned` or `number_accepted` counters.
- If client service has finished in a given cycle (`release_time` <= 0) and a new client is waiting in the queue, then the customer should be taken from the beginning of the queue. Member `release_time` should be set again based on the value of client's `service_time` (the window will be unavailable for this period of time). Remember to control the value of `number_served` (customer service has ended in this cycle);
- If a current client is being served in a given cycle, reduce the value of `release_time` by one;

Efekt poprawnego wykonania programu:

---------- PART 1 (1 pts) ----------
Client 1:
Arrival Time: 0 Service time: 0

Client 1:
Arrival Time: 3 Service time: 10

Client 2:
Arrival Time: 5 Service time: 10

---------- PART 2 (5 pts) ----------
Queue:
Empty queue!

Client 1 added to the queue
Client 2 added to the queue

Queue:
Arrival Time: 3 Service time: 10
Arrival Time: 5 Service time: 10

Client 1 has left the queue

Queue:
Arrival Time: 5 Service time: 10

Client 1:
Arrival Time: 3 Service time: 10

Client 2 has left the queue

Queue:
Empty queue!

Client 2:
Arrival Time: 3 Service time: 10

---------- PART 3 (1 pts) ----------
Window no 1 (queue size: 5, working time: 1h
The number of customers accepted into the queue: 0
The number of customers served by the window: 0
The number of clients returned from the queue: 0

Window no 2 (queue size: 10, working time: 1h
The number of customers accepted into the queue: 0
The number of customers served by the window: 0
The number of clients returned from the queue: 0

---------- PART 4 (1 pts) ----------

Window no 1 (queue size: 5, working time: 1h, client every: 5 minutes

The number of customers accepted into the queue: 11

The number of customers served by the window: 6

The number of clients returned from the queue: 1

Window no 2 (queue size: 10, working time: 1h, client every: 3 minutes

The number of customers accepted into the queue: 16

The number of customers served by the window: 6

The number of clients returned from the queue: 4