Приложение №1

```
#include <iostream>

#include <thread>

#include <vector>

#include <queue>

#include <mutex>

#include "semaphore.h"

#include <random>



/**

 * Выполнил: студент БПИ199 Кочарян Тигран Самвелович.

 * ----------------------------------------------------------------

 * Условие: Задача о супермаркете.

 * В супермаркете работают два кассира, покупатели заходят в супермаркет,

 * делают покупки и становятся в очередь к случайному кассиру.

 * Пока очередь пуста, кассир спит, как только появляется покупатель,

 * кассир просыпается. Покупатель спит в очереди, пока не подойдет к кассиру.

 * Создать многопоточное приложение, моделирующее рабочий день супермаркета.

 */



using namespace std;
```

```
// Информация о первой очереди: семафоры и мютексы данной очереди.

queue<sem_t*> first_queue;

sem_t first_cashier;

mutex first_queue_mutex;



// Информация о второй очереди: семафоры и мютексы данной очереди.

queue<sem_t*> second_queue;

sem_t second_cashier;

mutex second_queue_mutex;



// Максимальное число очередей задачи и мютекс на вывод информации в косноль.

static const int MAX_QUEUE_NUMS = 2;

static int clients_num;

mutex clients_num_mutex;

mutex output_mutex;



static mt19937 mt_rand(chrono::high_resolution_clock::now().time_since_epoch().count());
```

```
/**
 * Рандомный выбор между очередями.
 * @return 0 или 1.
 */
int choose_queue() {
	return mt_rand() % MAX_QUEUE_NUMS;
}


/**
 * Симуляция нахождения в очереди.
 * Будим кассира, спим, ждем, пока нас разбудят.
 * @param client_num
 */
void queue_simulation(int client_num) {
	// Инициализация семафора данного клиента в очереди.
	sem_t* sem = new sem_t;
	int queue_num = choose_queue();
	sem_init(sem, 0, 0);

	// Если 0 - то идем в первую очередь, если 1 - то во вторую.
	if (queue_num == 0) {
	first_queue_mutex.lock();
	first_queue.push(sem);
	sem_post(&first_cashier);
```

```
	first_queue_mutex.unlock();

	} else {

	second_queue_mutex.lock();

	second_queue.push(sem);

	sem_post(&second_cashier);

	second_queue_mutex.unlock();

	}



	// Ждем кассира. Как только нас будят, выводим результат.

	sem_wait(sem);

	output_mutex.lock();

	cout << "Cashier №" << queue_num+1 << " has served the Customer №" << client_num + 1 << "\n";

	output_mutex.unlock();

	clients_num_mutex.lock();

	clients_num--;

	clients_num_mutex.unlock();

}



/**

 * Симуляция работы первого кассира.

 * Пока кто-то есть в очереди, ждем, пока нас разбудят.

 */

void serve_first() {

	while (!first_queue.empty()) {
```

```
	sem_wait(&first_cashier);
	first_queue_mutex.lock();
	sem_post(first_queue.front());
	first_queue.pop();
	first_queue_mutex.unlock();
	}
	sem_post(&second_cashier);
}

/**
 * Симуляция работы второго кассира.
 * Пока кто-то есть в очереди, ждем, пока нас разбудят.
 */
void serve_second() {
	while (!second_queue.empty()) {
	sem_wait(&second_cashier);
	second_queue_mutex.lock();
	sem_post(second_queue.front());
	second_queue.pop();
	second_queue_mutex.unlock();
	}
	sem_post(&first_cashier);
}
```

```
/**
 * Вызов функций и создание потоков.
 * @return информацию об ошибках.
 */
int main() {
	srand(time(NULL));

	std::cout << "Please, input the number of customers:\n";
	std::cin >> clients_num;

	// Проверка на корректность ввода информации о кол-ве клиентов.
	if(clients_num == 0) {
	cout << "Just another chill day in da store..." << endl;
	exit(EXIT_SUCCESS);
	} else if (clients_num < 0) {
	cout << "Wrong Input Data" << endl;
	exit(EXIT_FAILURE);
	}

	// Инициализация семафора первой очереди.
	sem_init(&first_cashier, 0, 0);
	sem_init(&second_cashier, 0, 0);

	// Создание потоков клиентов, находящихся в очереди и обслуживающих продавцов.
```

```
        std::vector<std::thread> threads;
        for (int i = 0; i < clients_num; ++i) {
        threads.emplace_back(queue_simulation, i);
        }
        threads.emplace_back(serve_first);
        threads.emplace_back(serve_second);

        // Соединение потоков с основным потоком.
        for (int i = 0; i < threads.size(); ++i) {
        threads[i].join();
        }
}
```