


```
template<typename T>
class drow_queue_v2
{
public:
    drow_queue_v2 (size_t capacity_)
        : capacity (std::bit_ceil (capacity_))
    {}

    bool try_push (const T&);
    bool try_pop (T&);

private:
    size_t capacity = 0;
    std::vector<T> data { std::vector<T> (capacity) };
    std::atomic<size_t> head { 0 }, tail { 0 };
};
```

```
bool try_push (const T& v)
{
    size_t current_tail = tail.load();
    size_t current_head = head.load();

    size_t size = current_tail - current_head;

    if (size >= (capacity - 1)) // full
        return false;

    size_t index = current_tail & (capacity - 1);
    data[index] = v;
    tail.store (current_tail + 1);

    return true;
}
```

```
bool try_pop (T& v)
{
    size_t current_head = head.load();
    size_t current_tail = tail.load();

    if (current_head == current_tail) // empty
        return false;

    size_t index = current_head & (capacity - 1);
    v = data[index];
    head.store(current_head + 1);

    return true;
}
```







```

template<typename T>
class drow_queue_v2
{
public:
    drow_queue_v2 (size_t capacity_)
        : capacity (std::bit_ceil (capacity_))
    {}

    bool try_push (const T&);
    bool try_pop (T&);

private:
    size_t capacity = 0;
    std::vector<T> data { std::vector<T> (capacity) };
    std::atomic<size_t> head { 0 }, tail { 0 };
};

```

```

bool try_push (const T& v)
{
    size_t current_tail = tail.load();
    size_t current_head = head.load();

    size_t size = current_tail - current_head;

    if (size >= (capacity - 1)) // full
        return false;

    size_t index = current_tail & (capacity - 1);
    data[index] = v;
    tail.store (current_tail + 1);

    return true;
}

```

```

bool try_pop (T& v)
{
    size_t current_head = head.load();
    size_t current_tail = tail.load();

    if (current_head == current_tail) // empty
        return false;

    size_t index = current_head & (capacity - 1);
    v = data[index];
    head.store(current_head + 1);

    return true;
}

```


