```
#pragma once
#include <vector>
#include <stdexcept>

template<typename  T>
class RingBuffer {
//----------
public:
	RingBuffer(size_t size) : buffer(size), maxSize(size),
                                    head(0), tail(0), full(false) {}
	// データ追加(最大のとき、データは上書きする)
	void push(const T& item) {
		//TODO 04_01
		// データ登録
		// ヘッダにデータを保存
		buffer[head] = item;
		// ヘッダをずらす(最大値を越えた場合は循環するように)
		head = (head + 1) % maxSize;

		// バッファ領域が最大まで登録されている場合
		if (full) {
			// 取り出し位置をずらす。
			// ずらした位置が最大値を越えた場合は循環するように
			tail = (tail + 1) % maxSize;
		}
		// 書き出しと呼び出し位置が同じとき full のフラグを立てる
			full = head == tail;
	}

	// データ削除
	bool pop(T& item) {
		// TODO 04_02
		// データが空の場合 false を返す
		if (empty()) {
			return false;
		}
```

```
  // TODO 04_02 続き
  // 呼び出し位置からデータを取り出す
  item = buffer[tail];
  // 呼び出し位置をずらす(最大値を越えた場合は循環するように)
  tail = (tail + 1) % maxSize;
  // full フラグをおろす
  full = false;
  return true;
}//bool pop(T& item)
// インデックスを指定して参照
// 登録データの取り出しや最大値以上追加は参照位置が変わるので注意
T& at(size_t index) {
  // TODO 04_03
  // 空の場合、throw std::out_of_range("Buffer is empty");を実行
  if (empty()) {
        throw std::out_of_range("Buffer is empty");
    }
    //登録データ数を越えた値が指定された場合、
    //throw std::out_of_range("Index out of range");を実行
    size_t s = size();
    if (index >= size()) {
        throw  std::out_of_range("Index out of range");
    }
    // 取り出し位置からインデックスの値ずらした位置を計算
    //(最大値を越えた場合は循環するように)
    size_t actualIndex = (tail + index) % maxSize;
    return buffer.at(actualIndex);
}

bool empty() const {
  return (!full && (head == tail));
}
bool fullBuffer() const {
    return full;
}
```

```
        size_t size() const {
          size_t size = maxSize;

          if (!full) {
                if (head >= tail) {
                        size = head - tail;
                }
                else {
                        size = maxSize + head - tail;
                }
          }

          return size;
        }

        size_t capacity() const {
                return maxSize;
        }
//--------
private:
        std::vector<T> buffer;  // バッファ保存領域
        size_t maxSize;                 // 最大値
        size_t head;                    // 次のデータの書き込み先
        size_t tail;                    // データの取り出し先
        bool full;                              //  データがバッファ領域最大まで入っているか
};
```