Practical 7 page replacement

```
#include <iostream>
#include <unordered_set>
#include <queue>
class FIFO {
private:
    int capacity;
    std::queue<int> memory;
    std::unordered_set<int> pageSet;
public:
    FIFO(int capacity) : capacity(capacity) {}
    std::string pageFault(int page) {
        if (pageSet.find(page) == pageSet.end()) {
            if (memory.size() == capacity) {
                int evictedPage = memory.front();
                memory.pop();
                pageSet.erase(evictedPage);
            }
            memory.push(page);
            pageSet.insert(page);
            return "Fault";
        }
        return "Hit";
    }
};
int main() {
    FIFO fifo(3);
    std::cout << fifo.pageFault(1) << std::endl; // Fault
    std::cout << fifo.pageFault(2) << std::endl; // Fault
    std::cout << fifo.pageFault(3) << std::endl; // Fault
    std::cout << fifo.pageFault(1) << std::endl; // Hit
    std::cout << fifo.pageFault(4) << std::endl; // Fault
    return 0;
}
```

Output

```
/tmp/8FdV4rGtWS.o
Fault
Fault
Fault
Hit
Fault


=== Code Execution Successful ===
```