참치 통조림 관리

(주)준하시스템에서 근무하는 사 대리는 참치를 매우 좋아하는 직장 상사 무 팀장의 지시로 참치 통조림의 유통기한 관리 시스템을 개발하려고 합니다. 이 시스템은 매입한 참치의 유통기한을 저장하고, 참치의 출고 요청이 들어오면 오래된 순서대로 요청된 개수만큼의 참치 유통기한을 출력해야 합니다. 매입과 출고는 임의의 순서로 일어날 수 있습니다.

문제

- 입력받은 참치의 유통기한을 Heap으로 구현한 Priority Queue에 저장합니다.
- 출고 요청이 오면 요청 개수만큼 오래된(가장 작은 숫자의) 유통기한을 출력합니다.
- 0을 입력받으면 프로그램을 종료합니다.

- 참치는 최대 100개까지만 입력됩니다.
- 참치의 유통기한은 20200321과 같이 연월일 순서의 8자리의 정수로 입력됩니다. 이 숫자는 최소한 20200101 이후입니다.
- 참치의 출고 요청은 1~20까지 최대 2자리의 정수로 입력됩니다.
- 저장된 참치 수보다 많은 수의 출고 요청은 발생하지 않습니다. 별도의 처리를 구현하지 않아도 무방합니다.
- 음수는 입력되지 않습니다. 별도의 처리를 구현하지 않아도 무방합니다.
- 연월일 유효성 검사(월 값이 1~12인지, 일 값이 해당 월의 범위 안에 있는지) 또한 구현하지 않아도 무방합니다.

입출력 형식 (초록색 : 입력, 검은색 : 출력)

```
20220217
20201213
20200927
20240724
20211125
20200926
4
20200926
20200927
20201213
20211125
20221104
2
20220217
```

```
20221104
20240624
2
20240624
20240724
0
```

힌트

- 대소비교는 유통기한 숫자를 그대로 비교하면 됩니다.
- 작은 숫자의(유통기한이 가까운) 유통기한부터 출력해야 하므로, Min-Heap과 Max-Heap 중 적절한 형태의 Heap을 선정해야 합니다.
- 이 문제는 다음과 같은 과정으로 세분화할 수 있습니다. 어느 과정이 어느 시점에 실행되어야 하는지, 어느 과정의 무슨 데이터를 어떤 조건으로 검사해야 하고 어떻게 분기되어야 하는지 등을 순서도로 설계해보면 코딩 과정에서의 혼란을 예방할 수 있습니다.
  * Heap 구현 (가급적 별도의 클래스로 구현하는 것을 권장)
    * Heap 내부의 자료구조 구현 (배열을 권장)
    * Heap 입출력 함수 설계
    * Heapify 함수 구현, 호출 시점 선정
  * 메인 함수에서의 메인 루프 구현
    * 숫자 입력 구현
    * 조건 검사(0이면 종료, 1~20이면 출력, 20200101 이상이면 입력)
    * 0일 때 종료 구현 (break;)
    * 1~20일 때 해당 개수만큼 Heap에서 꺼내기 (반복문 구현)
    * 유통기한 숫자일 때 Heap에 해당 숫자를 넣기

모범답안

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

class minheap {
public:
    minheap()
     : capacity(100001), size(0), inf(2147483647) {
        arr = new int[capacity];
        for (int i = 0; i < capacity; i++) {
            arr[i] = inf;
        }
    }
    bool empty() {
        return size == 0 ? true : false;
    }
    int top() {
        return arr[1];
    }
    void push(int value) {
        arr[++size] = value;
        for (int i = size; i > 1; i /= 2) {
            if (arr[i] < arr[i / 2]) {
                swap(arr[i], arr[i / 2]);
            }
            else {
                break;
            }
        }
    }
    void pop() {
        arr[1] = arr[size];
        arr[size--] = inf;
        for (int i = 1; i * 2 <= size;) {
            if (arr[i] < arr[i * 2] && arr[i] < arr[i * 2 + 1]) {
                break;
            }
            if (arr[i * 2] < arr[i * 2 + 1]) {
                swap(arr[i], arr[i * 2]);
                i = i * 2;
            }
            else {
                swap(arr[i], arr[i * 2 + 1]);
                i = i * 2 + 1;
            }
        }
    }
private:
    int capacity;
    int size;
    int inf;
    int* arr;
};
```

```cpp
class maxheap {
public:
    maxheap()
        :capacity(100001), size(0), inf(-1) {
        arr = new int[capacity];
        for (int i = 0; i < capacity; i++) {
            arr[i] = -1;
        }
    }
    bool empty() {
        return size == 0 ? true : false;
    }
    int top() {
        return arr[1];
    }
    void push(int value) {
        arr[++size] = value;
        for (int i = size; i > 1; i /= 2) {
            if (arr[i] > arr[i / 2]) {
                swap(arr[i], arr[i / 2]);
            }
            else {
                break;
            }
        }
    }
    void pop() {
        arr[1] = arr[size];
        arr[size--] = inf;
        for (int i = 1; i * 2 <= size;) {
            if (arr[i] > arr[i * 2] && arr[i] > arr[i * 2 + 1]) {
                break;
            }
            if (arr[i * 2] < arr[i * 2 + 1]) {
                swap(arr[i], arr[i * 2 + 1]);
                i = i * 2 + 1;
            }
            else {
                swap(arr[i], arr[i * 2]);
                i = i * 2;
            }
        }
    }
private:
    int capacity;
    int size;
    int inf;
    int* arr;
};
```

```cpp
int main() {
    minheap pq;
    while (true) {
        int n; cin >> n;
        if (n > 20) {
            pq.push(n);
        }
        else if (n == 0) {
            break;
        }
        else {
            for (int i = 0; i < n; i++) {
                cout << pq.top() << "\n";
                pq.pop();
            }
        }
    }
    return 0;
}
```