#include<iostream>

using namespace std;

//class CircularQueue {

// int\* data;

// int capacity;

// int front = 0;

// int rear = -1;

// int count = 0;

//public:

// CircularQueue() = default;

// CircularQueue(int size)

// {

// data = new int[size] {};

// capacity = size;

// front = 0;

// rear = 0;

// count = 0;

// }

//

// void enqueue(int value) {

// data[rear] = value;

// ++rear;

// ++count;

// }

//

// void dequeue() {

// auto newarray = new int[size()]{};

// for (size\_t i = 0; i < size()-1; i++)

// {

// newarray[i] = data[i + 1];

// }

// newarray[size() - 1] = data[0];

// if (size() != 0) {

// delete[]data;

// }

// data = newarray;

// newarray = nullptr;

// }

// int peek()const {

// return data[front];

// }

// int size()const {

// return count;

// }

// bool isEmpty()const {

// return size() == 0;

// }

//

// bool isFull()const {

// return capacity == size();

// }

//

// void show()const {

// for (size\_t i = 0; i < size(); i++)

// {

// cout << data[i] << " ";

// }

// cout << endl;

// }

//

// ~CircularQueue()

// {

// delete[]data;

// }

//};

//

//

//void main() {

//

// CircularQueue queue(3);

// for (size\_t i = 0; i < 3; i++)

// {

// queue.enqueue(i + 1);

// }

// queue.show();

// cout << "Start" << endl;

// for (size\_t i = 0; i < 3; i++)

// {

// queue.dequeue();

// queue.show();

// }

//}

template<typename T>

class PriorityQueue {

T\* data;

int capacity;

int front = 0;

int rear = -1;

int count = 0;

public:

PriorityQueue() = default;

PriorityQueue(int size)

{

data = new T[size] {};

capacity = size;

front = 0;

rear = 0;

count = 0;

}

void enqueue(const T& value) {

data[rear] = value;

++rear;

++count;

}

void dequeue() {

int index = getIndexOfMax();

auto temp = new T[capacity] {};

for (size\_t i = 0; i < index; i++)

{

temp[i] = data[i];

}

for (size\_t i = index,i2=index+1; i < size()-1; i++,i2++)

{

temp[i] = data[i2];

}

if (size() != 0) {

delete[]data;

}

data = temp;

temp = nullptr;

--count;

--rear;

}

T peek()const {

return data[front];

}

int size()const {

return count;

}

bool isEmpty()const {

return size() == 0;

}

bool isFull()const {

return capacity == size();

}

void show()const {

for (size\_t i = 0; i < size(); i++)

{

cout << data[i] << " ";

}

cout << endl << endl;

}

int getIndexOfMax() {

T max = data[0];

int maxindex = -1;

for (size\_t i = 0; i < size(); i++)

{

if (data[i] >= max) {

max = data[i];

maxindex = i;

}

}

return maxindex;

}

~PriorityQueue()

{

delete[]data;

}

};

#include<Windows.h>

class Player {

string username;

double score;

public:

Player() = default;

Player(const string&u,const double&s)

{

this->username = u;

this->score = s;

}

bool operator>=(const Player& other) {

return this->score >= other.score;

}

friend ostream& operator<<(ostream& out, const Player& player);

};

ostream& operator<<(ostream& out, const Player& player) {

out << "Player Info" << endl;

out << "Username : " << player.username << endl;

out << "Score : " << player.score << endl;

return out;

}

template<typename T,int size>

class Container {

PriorityQueue<T>pqueue(size);

};

void main() {

PriorityQueue<Player> queue(5);

queue.enqueue(Player("Jordan",78));

queue.enqueue(Player("Lebron",90));

queue.enqueue(Player("Kyle Anderson",99));

queue.enqueue(Player("Carmelo Anthony",90));

while (!queue.isEmpty()) {

queue.show();

queue.dequeue();

Sleep(2000);

}

/\*PriorityQueue<int> queue(5);

queue.enqueue(-5);

queue.enqueue(5);

queue.enqueue(0);

queue.enqueue(7);

queue.enqueue(16);

while (!queue.isEmpty()) {

queue.show();

queue.dequeue();

Sleep(2000);

} \*/

}