CS302

Assignment 6: Mario Kart

Description

Mario Kart. The game that can ruin friendships, especially if you prime a red shell right behind your friend right before they're about to win the race. In this simulation of Mario Kart, there are no opportunities to pick up any items, just an ordinary race. Thus your racing skill will be tested, we will maintain the racers in the race using a priority queue (a binary min heap).

```
template <class t1, class t2>
class priorityQ
{
public:
  priorityQ();
  priorityQ(const priorityQ<t1, t2>&);
  const priorityQ<t1, t2>& operator=(const priorityQ<t1, t2>&);
  ~priorityQ();
  void push_back(const t1&, const t2&);
  void pop_front();
  void decrease_priority(const t1&, const t2&);
  t2 get_front_priority() const;
  t1 get_front_key() const;
  bool isEmpty() const;
private:
  struct priorityType
  {
    t1 key;
    t2 priority;
  };

  void bubbleUp(std::size_t);
  void bubbleDown(std::size_t);

  priorityType *  heapArray;
  std::size_t capacity, size;
  std::unordered_map<t1, std::size_t> itemToPQ;
};
```

Each member contains/performs the following

- `struct prioityType` - each element has a key id and its priority
- `priorityType * heapArray` - the list of elements that maintains the priority queue

- `size_t capacity` - the amount of elements `heapArray` can hold
- `size_t size` - the amount of elements that have been inserted into the priority queue
- `std::unordered_map<t1, std::size_t> itemToPQ;` - a map that maps a key to the index in the `heapArray`
- `priorityQ<t1, t2>::priorityQ()` - sets up an empty priority queue, if you wish to have the root start from index 1, then just have `heapArray.resize(1)` in the constructor's body otherwise the constructor's body would be empty
- `priorityQ<t1, t2>::priorityQ(const priorityQ<t1, t2>& copy)` - copy constructor that performs a deep copy of the object
- `const priorityQ<t1, t2>& priorityQ<t1, t2>::operator=(const priorityQ<t1, t2>& rhs)` - deep copy assignment operator
- `priorityQ<t1, t2>::~priorityQ()` - destructor
- `void priorityQ<t1, t2>::push_back(const t1& key, const t2& priority)` - inserts a new `priotityType` object to the back of `heapArray`, maps this key to the last index of the `heapArray` and then calls `bubbleUp()`, resize `heapArray` if needed (perhaps double its capacity)
- `void priorityQ<t1, t2>::pop_front()` - assigns the last element of `heapArray` and assigns it to the root position, updates the `itemToPQ` map accordingly, then call `bubbleDown(1)` or `bubbleDown(0)` (depending on whether you designate 1 or 0 as the root)
- `void priorityQ<t1, t2>::decrease_priority(const t1& key, const t2& priority)` - using the `itemToPQ` map, you update the value field of the element in the `heapArray` and then call `bubbleUp` using the index of where `key` parameter is in `heapArray`
- `t2 priorityQ<t1, t2>::get_front_priority() const` - returns the priority field of the root element
- `t1 priorityQ<t1, t2>::get_front_key() const` - returns the key field of the root element
- `bool priorityQ<t1, t2>::isEmpty() const` - returns `true` if there is nothing in the priority queue and `false` otherwise
- `void priorityQ<t1, t2>::bubbleUp(std::size_t index)` - function that performs the standard bubble up procedure, you compute the parent index (using the index parameter) and compare the parent and child's value fields, and swap if necessary, and continue up the tree until you cannot bubble up any further
- `void priorityQ<t1, t2>::bubbleDown(std::size_t index)` - function that performs the standard bubble down procedure, using the parameter index, compute the left and right children indices and compare their value fields and swap the parent with the correct child and continue until you cannot bubble down any further

Main

You read in contents from a file using input redirection. The file first contains a set of racer names (one name per line), once the string `"END"` is read, you have read all the racers names. Each racer is inserted into a `priorityQ<string, double>` object (the racer's name is the key and 50 will be the priority, each racer is 50 units away from the finish line)

The next set of lines simulate intervals of the race (one time unit), where in an interval each racer moves closer to the finish line. You need to read the racer's name and their new distance and then update that racer's position in the priority queue.

A racer cannot be updated more than once in an interval, so if you read in the same racer again in the interval, output `"Already updated <racer name> in this interval!"` and then read a racer's name again. The input file is formatted specifically where if a racer has already been updated and that racer's name appears again, you won't need to read in the updated distance.

You then would need to call `decrease_priority()` function to update that racer in the `priorityQ<string, double>` object and output `"Updating <racer name>"`. Each interval ends either with a line that contains `"DONE"` or when every racer that is still in the race was updated in the interval. Then you apply the same technique on a new interval. At the start of every interval you check if any racer has finished (if the leading racer has a distance of 0 or less), you pop the racer and output its rank.

During any interval, if a racer's name is read in, but that racer has already finished the race, you output `"Racer has already finished the race!"` and then just read the next racer's name. The simulation is over once all of the racers have finished the race.

The interval counter is to be updated at the start of each interval. You can also create `unordered_map` objects in main, why would you ever linear search a list ever again?

Specifications

- Use your own custom `priorityQ` object, and use this to maintain all the racers
- Do not modify the `priorityQ` class by adding new members
- Comment your code thoroughly
- Global variables are not permitted

1 Example Run

```
% g++ main.cpp
```

```
% ./a.out < simulation01.txt

Interval 1

Updating DonkeyKong
Updating Luigi
Updating Bowser
Updating Mario

Interval 2

Updating DONKEYKONG
Updating Bowser
Updating Yoshi
Updating peach
Updating Toad
Already updated DonkeyKong in this interval!

Updating Wario
Updating Luigi
Updating Mario

Interval 3

Updating Yoshi
Updating Bowser
Updating Wario
Already updated YOSHI in this interval!


Interval 4

Updating Yoshi
Updating Peach
Updating Wario
Updating Luigi

Interval 5

Updating Luigi
Updating Bowser

Interval 6

1 Luigi

2 Bowser

Updating Toad
Racer has already finished the race!


Interval 7
```

```
Updating DonkeyKong
Updating Yoshi
Updating Peach
Updating Wario
Already updated Yoshi in this interval!


Interval 8

3 Yoshi

4 Wario

Updating Toad

Interval 9

5 Toad

Updating DonkeyKong

Interval 10

6 DonkeyKong

Updating Peach

Interval 11

7 Peach

Racer has already finished the race!

Updating Mario

Interval 12

Updating Mario

Interval 13

8 Mario

% ./a.out < simulation02.txt

Interval 1

Updating Luigi
Updating Waluigi
Updating Link
Updating Peach
Updating Toad
Updating ShyGuy
Updating KoopaTroopa
```

```
Updating Daisy
Updating Bowser
Updating Yoshi
Updating Mario
Updating DiddyKong
Updating KingBoo
Updating Rosalina
Updating BowserJr
Updating Wario
Updating DonkeyKong

Interval 2

Updating Luigi
Updating Rosalina
Updating Wario
Updating Peach
Updating Toad
Updating Waluigi
Updating Bowser
Updating Daisy
Updating KoopaTroopa
Updating ShyGuy
Updating Mario
Updating DiddyKong
Updating KingBoo
Updating Yoshi

Interval 3

Updating Luigi
Updating DonkeyKong
Updating KingBoo
Updating Peach
Updating Rosalina
Updating Waluigi
Updating Bowser
Updating Yoshi
Updating KoopaTroopa
Updating DiddyKong
Updating ShyGuy

Interval 4

Updating Peach

Interval 5

Updating Peach
Updating Luigi
Updating Yoshi
Updating Mario
Updating Wario
Updating ShyGuy
```

```
Updating Waluigi
Updating KingBoo
Updating KoopaTroopa
Updating Rosalina
Updating DiddyKong
Updating Toad

Interval 6

Updating Link
Updating Bowser
Updating DiddyKong
Updating Mario
Updating Peach
Updating ShyGuy
Updating Waluigi
Updating Daisy
Updating KoopaTroopa
Updating Luigi
Updating Yoshi
Updating Toad
Updating BowserJr
Updating DonkeyKong
Updating Rosalina
Updating Wario

Interval 7

Updating DiddyKong
Updating Luigi
Updating KoopaTroopa
Updating Mario

Interval 8

Updating DiddyKong
Updating Mario
Updating Toad
Updating KingBoo
Updating Peach
Updating ShyGuy
Updating Wario
Updating Daisy
Updating Bowser

Interval 9

Updating Waluigi
Updating Mario
Updating KingBoo
Updating Luigi
Updating Peach
Updating DonkeyKong
Updating Wario
```

```
Updating Toad
Updating Bowser
Updating Yoshi

Interval 10

Updating DonkeyKong

Interval 11

Updating DonkeyKong
Updating Daisy
Updating Bowser
Updating Rosalina
Updating ShyGuy
Updating Wario
Updating Luigi
Updating KingBoo
Updating Waluigi
Updating Yoshi
Updating Link
Updating KoopaTroopa
Updating Peach
Updating DiddyKong
Updating BowserJr

Interval 12

1 DiddyKong

Updating Luigi
Updating Link
Updating Yoshi
Updating Rosalina
Updating Peach
Updating Wario
Updating DonkeyKong
Updating Toad
Updating Bowser
Updating ShyGuy
Updating Waluigi
Updating KoopaTroopa
Updating Daisy
Updating BowserJr
Updating Mario
Updating KingBoo

Interval 13

2 Mario

Updating ShyGuy
Updating BowserJr
Updating DonkeyKong
```

```
Updating Luigi
Updating Rosalina
Updating Link
Updating Yoshi
Updating Wario
Updating Bowser
Updating Peach
Updating Waluigi

Interval 14

3 Luigi

4 Peach

Updating BowserJr
Updating Waluigi
Updating DonkeyKong
Updating Bowser
Updating Link
Updating Toad
Updating KoopaTroopa

Interval 15

Updating KoopaTroopa
Updating Wario
Updating Toad

Interval 16

5 KoopaTroopa

Updating Link
Updating Rosalina
Updating Bowser
Updating KingBoo

Interval 17

Updating KingBoo
Updating Link
Updating Wario
Updating Bowser
Updating ShyGuy
Updating Daisy
Updating Waluigi
Updating Toad
Updating DonkeyKong
Updating Rosalina
Updating Yoshi
Updating BowserJr

Interval 18
```

```
6 Waluigi

7 Bowser

8 Toad

Updating ShyGuy
Updating Yoshi
Updating Wario
Updating BowserJr
Updating Link
Updating Rosalina
Updating DonkeyKong
Updating KingBoo
Updating Daisy

Interval 19

9 Yoshi

10 KingBoo

Updating BowserJr
Updating ShyGuy
Updating Wario

Interval 20

Updating BowserJr
Updating Link
Updating DonkeyKong
Updating Rosalina
Updating ShyGuy

Interval 21

11 DonkeyKong

12 ShyGuy

Updating Daisy
Updating Wario
Updating Link

Interval 22

13 Daisy

Updating Wario
Updating BowserJr

Interval 23
```

```
Updating BowserJr

Interval 24

Updating Rosalina

Interval 25

14 Rosalina

Updating Link

Interval 26

Updating BowserJr
Updating Link

Interval 27

Updating Link
Updating Wario

Interval 28

15 Wario

Updating BowserJr

Interval 29

Updating Link

Interval 30

16 Link

Updating BowserJr

Interval 31

Updating BowserJr

Interval 32

Updating BowserJr

Interval 33

Updating BowserJr

Interval 34

17 BowserJr
```

Submission

Submit the source file to code grade by the deadline

References

- Supplemental Video https://youtu.be/zv4m3NmedAI
- Link to the top image can be found at https://www.nicepng.com/downpng/u2t4u2i1q8r5t4e6_mario-kart-green-shells-red-shell/
- Link of patrick meme can be found at https://imgflip.com/memegenerator/348093413/We-Have-Technology