7️⃣

Priority Queue

In the case of the max heap, priority queues are a type of container adaptors, specifically designed such that its first element is always the greatest of the elements it contains and the rest elements are in decreasing order.

In the case of the min heap, priority queues are a type of container adaptors, specifically designed such that its first element is always the smallest of the elements it contains and the rest elements are in increasing order.

Note: In C++ STL by default max-heap is created.

The syntax for a max-heap priority queue

```
priority_queue<object_type> variable_name;
```

Example:

```
priority_queue<int> pq;
```

The syntax for a min-heap priority queue

```
priority_queue<object_type,vector<object_type>,greater<object_type>> variable_name;
```

Example:

```
priority_queue<int,vector<int>,greater<int>> pq;
```

The Whole Code:

```
// Containers --> Priority Queue
// Largest Element stay at the top
// Element not store in linear fashion it actually store in a tree
#include <bits/stdc++.h>
using namespace std;
void explainPriorityQueque()
{
      // Maximum Heap priority queue
      priority_queue<int> pq;
      pq.push(5); //{5}
      pq.push(2); // {5,2}
      pq.push(8); // {8,5,2}
      pq.emplace(10); // {10,8,5,2}

      cout << pq.top(); // print 10

      // Minimum Heap priority queue
      priority_queue<int,vector<int>,greater<int>> pq;
      pq.push(5); //{5}
      pq.push(2); // {2,5}
      pq.push(8); // {2,5,8}
      pq.emplace(10); // {2,5,8,10}

      cout << pq.top(); // print 2

}
int main()
{
      explainPriorityQueque();
      return 0;
}

/*
time complexity
```

```
push --> logn
top ---> O(1)
pop --> logn
*/
```

Functions in priority queue:

push() – to insert an element in the priority queue.

```
pq.push(110);
pq.push(220);
```

pop() – deletes the top element of the priority queue.

```
pq.pop();
```

top() – returns the element at the top of the priority queue.

```
pq.top();
```

emplace() – to insert an element in the priority.

```
pq.emplace(1);
pq.emplace(2);
```

size() – returns the number of elements in the priority queue.

```
pq.size();
```

empty() – to check if the priority queue is empty or not.

```
pq.empty();
```

Striver’s Code:

```
#include<bits/stdc++.h>
using namespace std;
void printpriorityqueue(priority_queue<int> pq)
{
    priority_queue<int> pq2=pq;
    while(!pq.empty())
    {
        cout<<pq.top()<<"\\n";
        pq.pop();
    }
}
int main()
{
    priority_queue<int> pq;
    for(int i=1;i<=5;i++)
    pq.push(i);

    cout<<"The elements of the priority queue are:"<<endl;
    printpriorityqueue(pq);

    cout<<"The size of the priority queue: "<<pq.size()<<endl;
    cout<<"The top element of the priority queue: "<<pq.top()<<endl;
    cout<<"Pop the top element: "<<endl;
    pq.pop();
    printpriorityqueue(pq);
}

Output:

The elements of the priority queue are:
5
4
3
2
1
The size of the priority queue: 5
The top element of the priority queue: 5
Pop the top element:
4
3
2
1
```