COMPARATOR CLASSES
APPLICATIONS OF PRIORITY QUEUES

STL Heap implementation: Priority Queues in C++

What is the output of this code?

```
priority_queue<int> pq;
pq.push(10);
pq.push(2);
pq.push(80);
cout<<pq.top();
pq.pop();
cout<<pq.top();
pq.pop();
cout<<pq.top();
pq.pop();
```

A. 10 2 80
B. 2 10 80
C. 80 10 2
D. 80 2 10
E. None of the above

Comparison class

- Comparison class: A class that implements a function operator for comparing objects

```
class compareClass{
      bool operator()(int& a, int & b) const {
            return a<b;
      }
};
```

Comparison class

```
class compareClass{
      bool operator()(int& a, int & b) const {
            return a<b;
      }
};


int main(){
    compareClass c;
    cout<<c(10, 20)<<endl;
}
```

```
What is the output of this code?
A.1
B.0
C.Error
```

STL Heap implementation: Priority Queues in C++

```
class compareClass{
      bool operator()(int& a, int & b) const {
            return a<b;
      }
};
 priority_queue<int, vector<int>, compareClass> pq;
 pq.push(10);
 pq.push(2);
 pq.push(80);
 cout<<pq.top();
 pq.pop();
 cout<<pq.top();
 pq.pop();
 cout<<pq.top();
 pq.pop();
```

This code prints the numbers in descending order: 80 10 2 (max-Heap)

How would you change it so that the top element is always the min value (min-Heap

std::priority_queue template arguments

```
template <
     class T,
     class Container= vector<T>,
     class Compare = less <T>
     > class priority_queue;
```

The template for priority_queue takes 3 arguments:

1. Type elements contained in the queue.
2. Container class used as the internal store for the priority_queue, the default is **vector<T>**
3. Class that provides priority comparisons, the default is **less**

std::priority_queue template arguments

```
//Template parameters for a max-heap
priority_queue<int, vector<int>, std::less<int>> pq;

//Template parameters for a min-heap
priority_queue<int, vector<int>, std::greater<int>> pq;
```

Application

Use priority queues to find the median of a sequence of numbers

Your implementation should allow for recomputing the median every time a new number is added to the sequence