COMPARISON CLASSES

Problem Solving with Computers-II

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;
```

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 cmp;
    cout<<cmp(10, 20)<<endl;
}
```

What is the output of this code?

A. 1

B. 0

C. Error

STL Heap implementation: Priority Queues in C++

```
class comparisonClass{
        bool operator()(int& a, int & b) const {
                return a>b;
        }
};
priority_queue<int, vector<int>, comparisonClass> 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();
```

pq is a _____heap

Sorting : Selection sort

```
//Precondition: unsorted vector v with N elements
//Post condition: sorted vector in ascending order
void selectionSort(vector<int>& v){
    int N=v.size();
    for(int i = 0; i < N; i++){
        int index=i;
        for(int j = i+1; j < N; j++){
            if(v[j] < v[index]){
                index = j;
            }
        }
        int tmp = v[i];
        v[i] = v[index];
        v[index] = tmp;
    }
}
```

```
10, 2, 80, 70, 50, 20
```

Sorting a forest (of Binary Search Trees)

```
//Precondition: unsorted vector v with N elements
//Post condition: sorted vector in ascending order
void selectionSort(vector<int>& v){
    int N=v.size();
    for(int i =0; i< N; i++){
        int index=i;
        for(int j = i+1; j<N;j++){
            if(v[j]<v[index]){
                index = j;
            }
        }
        int tmp = v[i];
        v[i] = v[index];
        v[index]=tmp;
    }
}
```

Modify selection sort to work with a forest: vector of bsts

Sort array elements using a pq storing pointers

```
int main(){
      int arr[]={10, 2, 80};
      priority_queue<int*> pq;
      for(int i=0; i < 3; i++)
            pq.push(arr+i);

      while(!pq.empty()){
            cout<<*pq.top()<<endl;
           pq.pop();
      }
      return 0;
}
```

How can we change the way pq prioritizes pointers?

Write a comparison class to print the integers in the array in sorted order

```
int main(){
      int arr[]={10, 2, 80};
      priority_queue<int*, vector<int*>, cmpPtr> pq;
      for(int i=0; i < 3; i++)
            pq.push(arr+i);

      while(!pq.empty()){
            cout<<*pq.top()<<endl;
           pq.pop();
      }
      return 0;
}
```