Exercise 13:

Submit your solutions (source code) to the questions below by email to your instructor and TA(s) by Monday, January 21st (16:30).

Question 1: Function objects (40 points).

This question will check your understanding of function objects (also called functors).

Your goal is to write a template function object named Bounded_min that can be "called" like a function of two arguments and returns the minimum of the two arguments; additionally this minimum value is bounded by the values of two parameters min_bound and max_bound specified to Bounded_min object via the constructor:

- Bounded_min(const T& min_bound, const T& max_bound)

First the minimum value of the two arguments is computed and returned unless it is outside of the specified bounds. If it is less than min_bound, then min_bound should be returned instead. If it is bigger than max_bound, then max_bound should be returned instead. Please write the code for Bounded_min in a file named "utils.h". To test this function, you can use the following code:

```
// test_Bounded_min.cpp
```

```
#include <iostream>
#include <cassert>
#include "utils.h"

using namespace std;

int main(void) {
  Bounded_min<int> b_min(0, 10);
  assert(b_min(1, 3) == 1);
  assert(b_min(-1, 15) == 0);
  assert(b_min(-1, -5) == 0);
  assert(b_min(12, 14) == 10);

  cout << "Tests passed" << endl;

  return 0;
}
```

Exercise 2: Function objects and container adapters (60 points).

The STL priority_queue is a queue in which elements are given a priority controlling which element goes to the top of the queue (the top contains the element with the highest priority). Priority is controlled by the function object Compare, which is one of the parametric types of the priority queue: template <class T, class Container = vector<T>, class Compare = <less<typename Container::value_type> > class priority_queue;

By default less (defined in the header <functional>) is used for comparisons. less in turn is defined using the operator '<' defined over the parametric type T. If we want to use a priority queue for a user defined object, we need therefore to overload operator< for our user defined object.

Sometimes, operator< may have already been specified but we may want to use a different behavior for prioritizing elements in the priority queue. In these cases, we can specify a function object to be used for comparing elements.

In the following code, a priority queue of ints is defined and used. The top-most element of the queue is always the biggest contained integer. Modify this code such that the top-most element contains always the smallest integer (of the queue). operator< is already defined for int, so you need in this case to specify a functor to be used for comparisons when defining the priority queue.

```
// max_priority_queue.cpp
#include<queue> // for priority_queue
#include<iostream>
#include<cassert>

using namespace std;

int main(void) {
 priority_queue<int> pq;
 // push some numbers on the pqueue
 pq.push(5);
 pq.push(7);
 pq.push(1);
 pq.push(2);
 pq.push(3);
 // elements are prioritized by operator<
 // so the biggest int should be on top
 assert(pq.top() == 7);

 // remove two times the top of the pqueue
 pq.pop(); // removes the top, i.e. 7
 pq.pop(); // removes the top, i.e. 5
 assert(pq.top() == 3);

 cout << "Tests passed" << endl;
 return 0;
}
```