```
#include <iostream>
#include <list>
#include <cstdlib>
#include <sys/time.h>

#include "pairs.h"

int        main();
void       insert(std::list<pairs_t>& plist, pairs_t pair);

int main()
{
        int                                num;
        std::list<int>                     list;
        std::list<int>::iterator           itr;
        std::list<int>::reverse_iterator   ritr;
        std::list<pairs_t>                 plist;

        struct timeval                     tp;

  // get time of day
  gettimeofday(&tp,NULL);

  // use microseconds as the seed
//srand((unsigned int)tp.tv_usec);
  srand((unsigned int)1337);

  // sort while inserting, thus maintaining list as priority queue
  std::cout << "Inserting:" << std::endl;
  for(int i=0;i<10;i++) {
    num = static_cast<int>((float)rand()/(float)RAND_MAX*100.0);
    std::cout << num << " ";
    for(itr = list.begin(); itr != list.end() && num > *itr; ++itr);
    list.insert(itr,num);
  }
  std::cout << std::endl;

  std::cout << "list (front to back):" << std::endl;
  for(std::list<int>::iterator itr = list.begin(); itr != list.end(); ++itr)
    std::cout << *itr << " ";
  std::cout << std::endl;

  std::cout << "list (back to front, using operator--):" << std::endl;
  for(std::list<int>::iterator itr = --list.end(); itr != --list.begin(); --itr)
    std::cout << *itr << " ";
  std::cout << std::endl;

  std::cout << "list (back to front, using reverse_iterator):" << std::endl;
  for(ritr = list.rbegin(); ritr != list.rend(); ++ritr)
    std::cout << *ritr << " ";
  std::cout << std::endl;

  ///////////

  insert(plist,pairs_t(45.0,'b'));
  insert(plist,pairs_t(37.2,'c'));
  insert(plist,pairs_t(42.6,'a'));
  insert(plist,pairs_t(53.7,'d'));

  std::cout << "list (front to back):" << std::endl;
  for(std::list<pairs_t>::iterator pitr = plist.begin(); pitr != plist.end(); ++pitr)
    std::cout << *pitr << " ";
```

std::list

```
  std::cout << std::endl;

  std::cout << "list (back to front):" << std::endl;
  for(std::list<pairs_t>::reverse_iterator pitr = plist.rbegin(); pitr != plist.rend();
 ++pitr)
    std::cout << *pitr << " ";
  std::cout << std::endl;
}

void insert(std::list<pairs_t>& plist, pairs_t pair)
{
        std::list<pairs_t>::iterator        pitr;

  // sort while inserting, thus maintaining list as priority queue
  std::cout << "Inserting: " << pair << std::endl;
  for(pitr = plist.begin(); pitr != plist.end() && pair > *pitr; ++pitr);
    plist.insert(pitr,pair);
}
```

std::list

```cpp
#include <iostream>

#include "pairs.h"

std::ostream& operator<<(std::ostream& s, const pairs_t& rhs)
{
  s << "('" << rhs.c << "'," << rhs.x << ")";

  return s;
}

pairs_t::pairs_t(const pairs_t& rhs)
{
  // this constructor creates the pairs_t object given a constant
  // reference to another pairs_t object (result is a copy of
  // the other object)
  c = rhs.c;
  x = rhs.x;
}

const pairs_t& pairs_t::operator=(const pairs_t& rhs)
{
  if(this == &rhs)      // standard alias test
    return *this;

  c = rhs.c;
  x = rhs.x;

  return *this;
}

bool pairs_t::operator<(const pairs_t& rhs)
{
  // a pairs_t is smaller than another pairs_t iff its character, c, is smaller
  return c < rhs.c;
}

bool pairs_t::operator>(const pairs_t& rhs)
{
  // a pairs_t is larger than another pairs_t iff its character, c, is larger
  return c > rhs.c;
}
```

std::list