```
// $Id: sortedlist.cpp,v 1.12 2019-02-07 13:54:19-08 - - $

//
// List insertion algorithm.
// Insert nodes into a singly-linked list using only operator<
// to form comparisons.  Do not insert elements that already
// exist.
//

#include <iostream>
#include <stdexcept>
#include <string>
using namespace std;

template <typename Type>
struct xless {
   bool operator() (const Type& left, const Type& right) const {
      return left < right;
   }
};

template <typename Type>
struct xgreater {
   bool operator() (const Type& left, const Type& right) const {
      return left > right;
   }
};

```

```

template <typename item_t, class less_t=xless<item_t>>
struct sorted_list {
   struct node {
      item_t item;
      node* link;
      node (const item_t& item_, node* link_):
                  item(item_),   link(link_) {
      }
   };
   less_t less;
   node* head = nullptr;

   sorted_list() = default; // Needed because default is suppressed.
   sorted_list (const sorted_list&) = delete;
   sorted_list& operator= (const sorted_list&) = delete;
   ~sorted_list();

   void insert (const item_t& newitem);
   item_t& front() { return head->item; }
   void pop_front();
};

template <typename item_t, class less_t>
sorted_list<item_t,less_t>::~sorted_list() {
   while (head != nullptr) pop_front();
}

template <typename item_t, class less_t>
void sorted_list<item_t,less_t>::insert (const item_t& newitem) {
   node** curr = &head;
   while (*curr != nullptr and less ((*curr)->item, newitem)) {
      curr = &(*curr)->link;
   }
   if (*curr == nullptr or less (newitem, (*curr)->item)) {
      *curr = new node (newitem, *curr);
   }
}

template <typename item_t, class less_t>
void sorted_list<item_t,less_t>::pop_front() {
   if (head == nullptr) throw underflow_error (__PRETTY_FUNCTION__);
   node* old = head;
   head = head->link;
   delete old;
}
```

```

template <typename item_t, class less_t=xless<item_t>>
void process (int argc, char** argv, const string& label) {
   sorted_list<string,less_t> list;
   for (char** argp = &argv[1]; argp != &argv[argc]; ++argp) {
      cout << label << ": Insert: " << *argp << endl;
      list.insert (*argp);
   }
   cout << endl;
   for (auto itor = list.head; itor != nullptr; itor = itor->link) {
      cout << label << ": Sorted: " << itor->item << endl;
   }
   cout << endl;
}

int main (int argc, char** argv) {
   process<string> (argc, argv, "Default");
   process<string,xgreater<string>> (argc, argv, "Greater");
   return 0;
}

/*
//TEST// alias grind='valgrind --leak-check=full --show-reachable=yes'
//TEST// grind --log-file=sortedlist.out.log \
//TEST//        sortedlist foo bar baz qux zxcvbnm asdfg qwerty \
//TEST//        bar baz foo qwerty hello hello 01234 56789 \
//TEST//        >sortedlist.out 2>&1
//TEST// mkpspdf sortedlist.ps sortedlist.cpp* sortedlist.out*
*/
```

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: starting sortedlist.cpp
checksource sortedlist.cpp
ident sortedlist.cpp
sortedlist.cpp:
     $Id: sortedlist.cpp,v 1.12 2019-02-07 13:54:19-08 - - $
cpplint.py.perl sortedlist.cpp
Done processing sortedlist.cpp
g++ -Wall -Wextra -Wpedantic -Wshadow -fdiagnostics-color=never -std=gnu++2a -Wold-style-cast -g -O0 sortedlist.cpp -o sortedlist -lm
rm -f sortedlist.o
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ mkc: finished sortedlist.cpp
```

```
Default: Insert: foo
Default: Insert: bar
Default: Insert: baz
Default: Insert: qux
Default: Insert: zxcvbnm
Default: Insert: asdfg
Default: Insert: qwerty
Default: Insert: bar
Default: Insert: baz
Default: Insert: foo
Default: Insert: qwerty
Default: Insert: hello
Default: Insert: hello
Default: Insert: 01234
Default: Insert: 56789

Default: Sorted: 01234
Default: Sorted: 56789
Default: Sorted: asdfg
Default: Sorted: bar
Default: Sorted: baz
Default: Sorted: foo
Default: Sorted: hello
Default: Sorted: qux
Default: Sorted: qwerty
Default: Sorted: zxcvbnm

Greater: Insert: foo
Greater: Insert: bar
Greater: Insert: baz
Greater: Insert: qux
Greater: Insert: zxcvbnm
Greater: Insert: asdfg
Greater: Insert: qwerty
Greater: Insert: bar
Greater: Insert: baz
Greater: Insert: foo
Greater: Insert: qwerty
Greater: Insert: hello
Greater: Insert: hello
Greater: Insert: 01234
Greater: Insert: 56789

Greater: Sorted: zxcvbnm
Greater: Sorted: qwerty
Greater: Sorted: qux
Greater: Sorted: hello
Greater: Sorted: foo
Greater: Sorted: baz
Greater: Sorted: bar
Greater: Sorted: asdfg
Greater: Sorted: 56789
Greater: Sorted: 01234

```

```
==29639== Memcheck, a memory error detector
==29639== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==29639== Using Valgrind-3.14.0 and LibVEX; rerun with -h for copyright info
==29639== Command: sortedlist foo bar baz qux zxcvbnm asdfg qwerty bar baz foo qwerty hello hello 01234 56789
==29639== Parent PID: 29638
==29639== 
==29639== 
==29639== HEAP SUMMARY:
==29639==     in use at exit: 0 bytes in 0 blocks
==29639==   total heap usage: 52 allocs, 52 frees, 1,264 bytes allocated
==29639== 
==29639== All heap blocks were freed -- no leaks are possible
==29639== 
==29639== For counts of detected and suppressed errors, rerun with: -v
==29639== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```