```
 1:
 2: #ifndef __COMPAREX_H__
 3: #define __COMPAREX_H__
 4:
 5: #include "trace.h"
 6:
 7: //
 8: // We assume that the type type_t has a compare member function.
 9: //
10:
11: template <typename type_t>
12: struct comparex {
13:    int operator() (const type_t &left, const type_t &right) const;
14: };
15:
16: RCSH(__comparex_h__,
17: "$Id: comparex.h,v 1.5 2010-02-12 17:35:44-08 - - $")
18:
19: #endif
20:
```

```
 1:
 2: #ifndef __LISTMAP_H__
 3: #define __LISTMAP_H__
 4:
 5: #include "comparex.h"
 6: #include "pairx.h"
 7:
 8: template <typename key_t, typename value_t>
 9: class listmap {
10:    private:
11:       struct node {
12:          pairx <key_t, value_t> pair;
13:          node *prev;
14:          node *next;
15:          node (const pairx <key_t, value_t> &);
16:       };
17:       node *head;
18:       node *tail;
19:    public:
20:       typedef pairx <key_t, value_t> mappairx;
21:       class iterator {
22:          friend class listmap;
23:          public:
24:             mappairx &operator* ();
25:             mappairx *operator-> ();
26:             iterator &operator++ ();
27:             iterator &operator-- ();
28:             bool operator== (const iterator &) const;
29:             bool operator!= (const iterator &) const;
30:             void erase ();
31:             iterator ();
32:          private:
33:             iterator (listmap *map, node *where);
34:             node *where;
35:             listmap *map;
36:       };
37:       listmap ();
38:       listmap (const listmap &);
39:       listmap &operator= (const listmap &);
40:       ~listmap ();
41:       void insert (const pairx <key_t, value_t> &);
42:       iterator find (const key_t &) const;
43:       iterator begin ();
44:       iterator end ();
45:       bool empty () const;
46: };
47:
48: RCSH(__listmap_h__,
49: "$Id: listmap.h,v 1.6 2010-02-18 16:13:44-08 - - $")
50:
51: #endif
52:
```

```
 1:
 2: #ifndef __PAIRX_H__
 3: #define __PAIRX_H__
 4:
 5: #include "trace.h"
 6:
 7: //
 8: // Class pairx works like pair(3c++).
 9: //
10:
11: template <typename first_t, typename second_t>
12: struct pairx {
13:    pairx (const first_t &, const second_t &);
14:    first_t first;
15:    second_t second;
16: };
17:
18: template <typename first_t, typename second_t>
19: ostream &operator<< (ostream &, const pairx <first_t, second_t> &);
20:
21: //
22: // The following implicitly generated members will work,
23: // because they just send messages to the first and second
24: // fields, respectively.
25: //
26: // Caution:  pairx() does not initialize its fields unless
27: // first_t and second_t do so with their default ctors.
28: //
29: // pairx ();
30: // pairx (const pairx &);
31: // pairx &operator= (const pairx &);
32: // ~pairx ();
33: //
34:
35: RCSH(__pairx_h__,
36: "$Id: pairx.h,v 1.4 2010-02-11 20:28:50-08 - - $")
37:
38: #endif
39:
```

```
 1:
 2: #ifndef __TRACE_H__
 3: #define __TRACE_H__
 4:
 5: #include <iostream>
 6: #include <string>
 7: #include <vector>
 8:
 9: using namespace std;
10:
11: //
12: // traceflags -
13: //    static class for maintaining global trace flags, each indicated
14: //    by a single character.
15: // setflags -
16: //    Takes a string argument, and sets a flag for each char in the
17: //    string.  As a special case, '@', sets all flags.
18: // getflag -
19: //    Used by the TRACE macro to check to see if a flag has been set.
20: //    Not to be called by user code.
21: //
22:
23: class traceflags {
24:    private:
25:       static vector<char> flags;
26:    public:
27:       static void setflags (const string &optflags);
28:       static bool getflag (char flag);
29: };
30:
31: //
32: // TRACE -
33: //    Macro which expands into trace code.  First argument is a
34: //    trace flag char, second argument is output code that can
35: //    be sandwiched between <<.  Beware of operator precedence.
36: //    Example:
37: //       TRACE ('u', "foo = " << foo);
38: //    will print two words and a newline if flag 'u' is  on.
39: //    Traces are preceded by filename, line number, and function.
40: //
41:
42: #define TRACE(FLAG,CODE) { \
43:            if (traceflags::getflag (FLAG)) { \
44:               cerr << __FILE__ << ":" << __LINE__ << ":" \
45:                    << __func__ << ": "; \
46:               cerr << CODE << endl; \
47:            } \
48:         }
49:
50: //
51: // RCSH, RCSC -
52: //    Macros which allow RCS Id information to transfer to object
53: //    files and executable binaries.
54: //
55:
56: #define RCSH(NAME,ID) \
57: static const char __RCS_##NAME[] = "\0" ID;
58: #define RCSC(NAME,ID) \
59: static const char __RCS_C_##NAME[] = "\0" ID \
60: "\0$Compiled: " __FILE__ " " __DATE__ " " __TIME__ " $";
61:
62: RCSH(__trace_h__,
63: "$Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $")
64:
```

```
65: #endif
66:
```

```
 1: //
 2: // util -
 3: //    A utility class to provide various services not conveniently
 4: //    included in other modules.
 5: //
 6:
 7: #ifndef __UTIL_H__
 8: #define __UTIL_H__
 9:
10: #include <iostream>
11: #include <list>
12: #include <string>
13:
14: #ifdef __GNUC__
15: #include <stdexcept>
16: #endif
17:
18: using namespace std;
19:
20: #include "trace.h"
21:
22: //
23: // sys_info -
24: //    Keep track of execname and exit status.  Must be initialized
25: //    as the first thing done inside main.  Main should call:
26: //       sys_info::set_execname (argv[0]);
27: //    before anything else.
28: //
29:
30: class sys_info {
31:    public:
32:       static const string &get_execname ();
33:       static void set_exit_status (int status);
34:       static int get_exit_status ();
35:    private:
36:       friend int main (int argc, char **argv);
37:       static void set_execname (const string &argv0);
38:       static string *execname;
39:       static int exit_status;
40: };
41:
42: //
43: // datestring -
44: //    Return the current date, as printed by date(1).
45: //
46:
47: const string datestring ();
48:
49: //
50: // split -
51: //    Split a string into a list<string>..  Any sequence
52: //    of chars in the delimiter string is used as a separator.  To
53: //    Split a pathname, use "/".  To split a shell command, use " ".
54: //
55:
56: list<string> split (const string &line, const string &delimiter);
57:
58: //
59: // complain -
60: //    Used for starting error messages.  Sets the exit status to
61: //    EXIT_FAILURE, writes the program name to cerr, and then
62: //    returns the cerr ostream.  Example:
63: //       complain() << filename << ": some problem" << endl;
64: //
```

```
 65:
 66: ostream &complain();
 67:
 68: //
 69: // syscall_error -
 70: //    Complain about a failed system call.  Argument is the name
 71: //    of the object causing trouble.  The extern errno must contain
 72: //    the reason for the problem.
 73: //
 74:
 75: void syscall_error (const string &);
 76:
 77: //
 78: // operator<< (list) -
 79: //    An overloaded template operator which allows lists to be
 80: //    printed out as a single operator, each element separated from
 81: //    the next with spaces.  The item_t must have an output operator
 82: //    defined for it.
 83: //
 84:
 85: template <typename item_t>
 86: ostream &operator<< (ostream &out, const list<item_t> &vec);
 87:
 88: //
 89: // string to_string (thing) -
 90: //    Convert anything into a string if it has an ostream<< operator.
 91: //
 92:
 93: template <typename item_t>
 94: string to_string (const item_t &);
 95:
 96: //
 97: // thing from_string (cons string &) -
 98: //    Scan a string for something if it has an istream>> operator.
 99: //
100:
101: template <typename item_t>
102: item_t from_string (const string &);
103:
104: //
105: // Put the RCS Id string in the object file.
106: //
107:
108: RCSH(__util_h__,
109: "$Id: util.h,v 1.3 2010-02-12 17:35:44-08 - - $")
110:
111: #endif
112:
```

```
 1:
 2: #include <cstdlib>
 3: #include <iostream>
 4: #include <string>
 5:
 6: using namespace std;
 7:
 8: #include "listmap.h"
 9: #include "pairx.h"
10: #include "util.h"
11:
12: void scan_options (int argc, char **argv) {
13:    opterr = 0;
14:    for (;;) {
15:       int option = getopt (argc, argv, "@:");
16:       if (option == EOF) break;
17:       switch (option) {
18:          case '@':
19:             traceflags::setflags (optarg);
20:             break;
21:          default:
22:             complain() << "-" << (char) optopt << ": invalid option"
23:                        << endl;
24:             break;
25:       }
26:    }
27: }
28:
29: int main (int argc, char **argv) {
30:    sys_info::set_execname (argv[0]);
31:    scan_options (argc, argv);
32:
33:    listmap <string, string> test;
34:    for (int argi = 0; argi < argc; ++argi) {
35:       pairx <string, string> pair (argv[argi], to_string <int> (argi));
36:       cout << "Before insert: " << pair << endl;
37:       test.insert (pair);
38:    }
39:
40:    listmap <string, string>::iterator itor = test.begin();
41:    listmap <string, string>::iterator end = test.end();
42:    for (; itor != end; ++itor) {
43:       cout << "During iteration: " << *itor << endl;
44:    }
45:
46:    cout << "EXIT_SUCCESS" << endl;
47:    return EXIT_SUCCESS;
48: }
49:
50: RCSC(__main_cc__,
51: "$Id: main.cc,v 1.6 2010-02-18 16:13:44-08 - - $")
52:
```

```
 1:
 2: #include <string>
 3: using namespace std;
 4:
 5: #include "comparex.h"
 6: #include "trace.h"
 7:
 8: template <typename type_t>
 9: int comparex<type_t>::operator() (const type_t &left,
10:                                   const type_t &right) const {
11:    int result = left.compare (right);
12:    TRACE ('c', "compare (" << left << ", " << right
13:           << ") = " << result << endl);
14:    return result;
15: }
16:
17: #include "comparex.ccti"
18:
19: RCSC(__comparex_cc__,
20: "$Id: comparex.cc,v 1.7 2010-02-18 20:36:31-08 - - $")
21:
```

```
 1:
 2: #include "listmap.h"
 3:
 4: comparex <string> compare;
 5:
 6: template <typename key_t, typename value_t>
 7: listmap <key_t, value_t>::node::node (const mappairx &pair):
 8:             pair(pair), prev(NULL), next(NULL) {
 9: }
10:
11: template <typename key_t, typename value_t>
12: listmap <key_t, value_t>::listmap (): head(NULL), tail (NULL) {
13: }
14:
15: template <typename key_t, typename value_t>
16: listmap <key_t, value_t>::~listmap () {
17:    TRACE ('l', (void*) this);
18:    //iterator itor = begin ();
19:    //iterator iend = end ();
20:    //while (itor != iend) itor.erase ();
21: }
22:
23: template <typename key_t, typename value_t>
24: pairx <key_t, value_t> &
25: listmap <key_t, value_t>::iterator::operator* () {
26:    TRACE ('l', where->pair);
27:    return where->pair;
28: }
29:
30: template <typename key_t, typename value_t>
31: pairx <key_t, value_t> *
32: listmap <key_t, value_t>::iterator::operator-> () {
33:    TRACE ('l', where->pair);
34:    return &(where->pair);
35: }
36:
37: template <typename key_t, typename value_t>
38: typename listmap <key_t, value_t>::iterator &
39: listmap <key_t, value_t>::iterator::operator++ () {
40:    TRACE ('l', "First: " << map << ", " << where);
41:    TRACE ('l', "Second: " << map->head << ", " << map->tail);
42:    where = where->next;
43:    return *this;
44: }
45:
46: template <typename key_t, typename value_t>
47: typename listmap <key_t, value_t>::iterator &
48: listmap <key_t, value_t>::iterator::operator-- () {
49:    where = where->prev;
50:    return *this;
51: }
52:
53: template <typename key_t, typename value_t>
54: bool listmap <key_t, value_t>::iterator::operator==
55:             (const iterator &that) const {
56:    return this->where == that.where;
57: }
58:
59: template <typename key_t, typename value_t>
60: bool listmap <key_t, value_t>::iterator::operator!=
61:             (const iterator &that) const {
62:    return this->where != that.where;
63: }
64:
```

```
 65: template <typename key_t, typename value_t>
 66: listmap <key_t, value_t>::iterator::iterator ():
 67:             map (NULL), where (NULL){
 68: }
 69:
 70: template <typename key_t, typename value_t>
 71: listmap <key_t, value_t>::iterator::iterator (listmap *map,
 72:             node *where): map (map), where (where){
 73: }
 74:
 75: template <typename key_t, typename value_t>
 76: typename listmap <key_t, value_t>::iterator
 77: listmap <key_t, value_t>::begin () {
 78:    return iterator (this, head);
 79: }
 80:
 81: template <typename key_t, typename value_t>
 82: typename listmap <key_t, value_t>::iterator
 83: listmap <key_t, value_t>::end () {
 84:    return iterator (this, NULL);
 85: }
 86:
 87: template <typename key_t, typename value_t>
 88: bool listmap <key_t, value_t>::empty () const {
 89:    return head == NULL;
 90: }
 91:
 92: template <typename key_t, typename value_t>
 93: void listmap <key_t, value_t>::iterator::erase () {
 94: }
 95:
 96: template <typename key_t, typename value_t>
 97: void listmap<key_t, value_t>::insert
 98:             (const pairx <key_t, value_t> &pair) {
 99:    node *tmp = new node (pair);
100:    if (empty ()) {
101:       head = tail = tmp;
102:    }else {
103:       int cmp = compare (tail->pair.first, pair.first);
104:       cout << tail->pair.first << " cmp " << pair.first << " = "
105:            << cmp << endl;
106:       tail->next = tmp;
107:       tmp->prev = tail;
108:       tail = tmp;
109:    }
110:    TRACE ('l', &pair << "->" << pair);
111: }
112:
113: #include "listmap.ccti"
114:
115: //RCSC(__listmap_cc__,
116: //"$Id: listmap.cc,v 1.10 2010-02-18 16:13:44-08 - - $")
117:
```

```
 1:
 2: #include <iostream>
 3: #include <string>
 4:
 5: using namespace std;
 6:
 7: #include "pairx.h"
 8: #include "trace.h"
 9:
10: template <typename first_t, typename second_t>
11: pairx <first_t, second_t>::pairx
12:             (const first_t &thefirst, const second_t &thesecond):
13:             first (thefirst), second (thesecond) {
14:    TRACE ('p', *this);
15: }
16:
17: template <typename first_t, typename second_t>
18: ostream &operator<< (ostream &out,
19:                      const pairx <first_t, second_t> &that) {
20:    out << "[" << that.first << "," << that.second << "]";
21:    return out;
22: }
23:
24: #include "pairx.ccti"
25:
26: RCSC(__pairx_cc__,
27: "$Id: pairx.cc,v 1.6 2010-02-12 15:03:23-08 - - $")
28:
```

```
 1:
 2: #include <climits>
 3: #include <iostream>
 4: #include <limits>
 5: #include <vector>
 6:
 7: using namespace std;
 8:
 9: #include "trace.h"
10:
11: //
12: // ** BUG IN STL ** BUG IN STL **
13: // We should use vector<bool> instead of vector<char>,
14: // but vector<bool> has a bug:
15: // http://forums.sun.com/thread.jspa?threadID=5277939
16: // Static linking works, but doubles the size of the executable
17: // image.
18: // ** BUG IN STL ** BUG IN STL **
19: //
20:
21: typedef vector<char> boolvec;
22: boolvec traceflags::flags (UCHAR_MAX + 1, false);
23: const boolvec trueflags (UCHAR_MAX + 1, true);
24:
25: void traceflags::setflags (const string &optflags) {
26:    string::const_iterator itor = optflags.begin();
27:    string::const_iterator end = optflags.end();
28:    for (; itor != end; ++itor) {
29:       if (*itor == '@') {
30:          flags = trueflags;
31:       }else {
32:          flags[*itor] = true;
33:       }
34:    }
35:    // Note that TRACE can trace setflags.
36:    TRACE ('t',  "optflags = " << optflags);
37: }
38:
39: //
40: // getflag -
41: //    Check to see if a certain flag is on.
42: //
43:
44: bool traceflags::getflag (char flag) {
45:    // Bug alert:
46:    // Don't TRACE this function or the stack will blow up.
47:    bool result = flags[flag];
48:    return result;
49: }
50:
51: RCSC(__trace_cc__,
52: "$Id: trace.cc,v 1.1 2010-02-09 20:25:52-08 - - $")
53:
```

```
 1:
 2: #include <cerrno>
 3: #include <cstdlib>
 4: #include <cstring>
 5: #include <ctime>
 6: #include <sstream>
 7: #include <stdexcept>
 8: #include <string>
 9: #include <typeinfo>
10:
11: using namespace std;
12:
13: #include "util.h"
14:
15: int sys_info::exit_status = EXIT_SUCCESS;
16: string *sys_info::execname = NULL; // Must be initialized from main().
17:
18: void sys_info_error (const string &condition) {
19:    throw logic_error ("main() has " + condition
20:                + " called sys_info::set_execname()");
21: }
22:
23: void sys_info::set_execname (const string &argv0) {
24:    if (execname != NULL) sys_info_error ("already");
25:    int slashpos = argv0.find_last_of ('/') + 1;
26:    execname = new string (argv0.substr (slashpos));
27:    cout << boolalpha;
28:    cerr << boolalpha;
29:    TRACE ('u', "execname = " << execname);
30: }
31:
32: const string &sys_info::get_execname () {
33:    if (execname == NULL) sys_info_error ("not yet");
34:    return *execname;
35: }
36:
37: void sys_info::set_exit_status (int status) {
38:    if (execname == NULL) sys_info_error ("not yet");
39:    exit_status = status;
40: }
41:
42: int sys_info::get_exit_status () {
43:    if (execname == NULL) sys_info_error ("not yet");
44:    return exit_status;
45: }
46:
47: const string datestring () {
48:    time_t clock = time (NULL);
49:    struct tm *tm_ptr = localtime (&clock);
50:    char timebuf[128];
51:    size_t bufsize = strftime (timebuf, sizeof timebuf,
52:           "%a %b %e %H:%M:%S %Z %Y", tm_ptr);
53:    return timebuf;
54: }
55:
56: list<string> split (const string &line, const string &delimiters) {
57:    list<string> words;
58:    int end = 0;
59:    // Loop over the string, splitting out words, and for each word
60:    // thus found, append it to the output list<string>.
61:    for (;;) {
62:       int start = line.find_first_not_of (delimiters, end);
63:       if (start == string::npos) break;
64:       end = line.find_first_of (delimiters, start);
```

```
 65:       words.push_back (line.substr (start, end - start));
 66:    }
 67:    TRACE ('u', words);
 68:    return words;
 69: }
 70:
 71: ostream &complain() {
 72:    sys_info::set_exit_status (EXIT_FAILURE);
 73:    cerr << sys_info::get_execname () << ": ";
 74:    return cerr;
 75: }
 76:
 77: void syscall_error (const string &object) {
 78:    complain() << object << ": " << strerror (errno) << endl;
 79: }
 80:
 81: template <typename item_t>
 82: ostream &operator<< (ostream &out, const list<item_t> &vec) {
 83:    typename list<item_t>::const_iterator itor = vec.begin();
 84:    typename list<item_t>::const_iterator end = vec.end();
 85:    // If the list is empty, do nothing.
 86:    if (itor != end) {
 87:       // Print out the first element without a space.
 88:       out << *itor++;
 89:       // Print out the rest of the elements each preceded by a space.
 90:       while (itor != end) out << " " << *itor++;
 91:    }
 92:    return out;
 93: }
 94:
 95: template <typename type_t>
 96: string to_string (const type_t &that) {
 97:    ostringstream stream;
 98:    stream << that;
 99:    return stream.str ();
100: }
101:
102: template <typename type_t>
103: type_t from_string (const string &that) {
104:    stringstream stream;
105:    stream << that;
106:    type_t result;
107:    if ( !(stream >> result   // Can we read type from string?
108:        && stream >> std::ws  // Flush trailing white space.
109:        && stream.eof ())     // Must now be at end of stream.
110:    ) {
111:       throw domain_error (string (typeid (type_t).name ())
112:             + " from_string (" + that + ")");
113:    }
114:    return result;
115: }
116:
117: #include "util.ccti"
118:
119: RCSC(__util_cc__,
120: "$Id: util.cc,v 1.3 2010-02-12 15:03:23-08 - - $")
121:
```

```
 1:
 2: //
 3: // Template instantiation that g++ can not perform automatically.
 4: // This is not a header file and is incuded from listmap.template.
 5: // This is unfortunate and causes excessive coupling between
 6: // modules.
 7: //
 8:
 9: template int comparex<string>::operator() (const string &,
10:                                            const string &) const;
11:
12: RCSC(__comparex_ccti__,
13: "$Id: comparex.ccti,v 1.2 2010-02-18 20:38:40-08 - - $")
14:
```

```
 1:
 2: //
 3: // Template instantiation that g++ can not perform automatically.
 4: // This is not a header file and is incuded from listmap.template.
 5: // This is unfortunate and causes excessive coupling between
 6: // modules.
 7: //
 8:
 9: typedef listmap <string, string> listmap_ss;
10: typedef listmap <string, string>::iterator listmap_ss_itor;
11: typedef listmap <string, string>::mappairx listmap_ss_pair;
12:
13: template listmap_ss::listmap ();
14: template listmap_ss::~listmap ();
15: template void listmap_ss::insert (const pairx <string, string> &);
16: template listmap_ss_itor listmap_ss::begin ();
17: template listmap_ss_itor listmap_ss::end ();
18:
19: template listmap_ss_pair &listmap_ss_itor::operator* ();
20: template listmap_ss_pair *listmap_ss_itor::operator-> ();
21: template listmap_ss_itor &listmap_ss_itor::operator++ ();
22: template listmap_ss_itor &listmap_ss_itor::operator-- ();
23: template bool listmap_ss_itor::operator== (const iterator &) const;
24: template bool listmap_ss_itor::operator!= (const iterator &) const;
25: template void listmap_ss_itor::erase ();
26:
27: RCSC(__listmap_ccti__,
28: "$Id: listmap.ccti,v 1.5 2010-02-18 20:38:40-08 - - $")
29:
```

```
 1:
 2: //
 3: // Template instantiation that g++ can not perform automatically.
 4: // This is not a header file and is incuded from listmap.template.
 5: // This is unfortunate and causes excessive coupling between
 6: // modules.
 7: //
 8:
 9: template pairx <string, string>::pairx (const string &, const string &);
10: template ostream &operator<< (ostream &out,
11:          const pairx <string, string> &);
12:
13: RCSC(__pairx_ccti__,
14: "$Id: pairx.ccti,v 1.3 2010-02-18 20:38:40-08 - - $")
15:
```

```
 1: //
 2: // Template instantiation that g++ can not perform automatically.
 3: // This is not a header file and is incuded from listmap.template.
 4: // This is unfortunate and causes excessive coupling between
 5: // modules.
 6: //
 7:
 8: template string to_string <int> (const int &);
 9:
10: RCSC(__util_ccti__,
11: "$Id: util.ccti,v 1.3 2010-02-18 20:38:40-08 - - $")
```

```
 1: # $Id: Makefile,v 1.7 2010-02-18 20:36:31-08 - - $
 2:
 3: MKFILE      = Makefile
 4: DEPSFILE    = ${MKFILE}.deps
 5: NOINCL      = ci clean spotless
 6: NEEDINCL    = ${filter ${NOINCL}, ${MAKECMDGOALS}}
 7: GMAKE       = ${MAKE} --no-print-directory
 8: UNAME      ?= ${shell uname -s}
 9:
10: COMPILECCC  = CC -g -features=extensions
11: MAKEDEPSCCC = CC -xM1
12: ifeq (${CCC},g++)
13: COMPILECCC  = g++ -g
14: MAKEDEPSCCC = g++ -MM
15: endif
16:
17: CCHEADER    = comparex.h listmap.h pairx.h trace.h util.h
18: CCSOURCE    = main.cc ${CCHEADER:.h=.cc}
19: CCTEMPLATES = comparex.ccti listmap.ccti pairx.ccti util.ccti
20: EXECBIN     = keyvalue
21: OBJECTS     = ${CCSOURCE:.cc=.o}
22: OTHERS      = ${MKFILE} README
23: ALLSOURCES  = ${CCHEADER} ${CCSOURCE} ${CCTEMPLATES} ${OTHERS}
24: LISTFILES   = ${ALLSOURCES} ${DEPSFILE} Idents
25:
26: LISTING     = ../asg4-listmap.code.ps
27: CLASS       = cmps109-wm.w09
28: PROJECT     = asg3
29:
30: all : ${EXECBIN}
31:         @ echo Compiled with ${COMPILECCC}.
32:
33: ${EXECBIN} : ${OBJECTS}
34:         ${COMPILECCC} -o $@ ${OBJECTS}
35:
36: %.o : %.cc
37:         ${COMPILECCC} -c $<
38:
39: ci : ${ALLSOURCES}
40:         @ - checksource ${ALLSOURCES}
41:         cid + ${ALLSOURCES}
42:
43: lis : ${ALLSOURCES}
44:         ${GMAKE} idents >Idents
45:         mkpspdf ${LISTING} ${LISTFILES}
46:         - rm Idents
47:
48: clean :
49:         - rm ${OBJECTS} ${DEPSFILE} core ${EXECBIN}.errs
50:
51: spotless : clean
52:         - rm ${EXECBIN}
53:
54: submit : ${ALLSOURCES}
55:         - checksource ${ALLSOURCES}
56:         submit ${CLASS} ${PROJECT} ${ALLSOURCES}
57:         testsubmit ${CLASS} ${PROJECT} ${ALLSOURCES}
58:
59: deps : ${CCSOURCE} ${CCHEADER}
60:         @ echo "# ${DEPSFILE} created `LC_TIME=C date`" >${DEPSFILE}
61:         ${MAKEDEPSCCC} ${CCSOURCE} | sort | uniq >>${DEPSFILE}
62:
63: ${DEPSFILE} :
64:         @ touch ${DEPSFILE}
```

```
65:         ${GMAKE} deps
66:
67: idents : ${ALLSOURCES} ${OBJECTS} ${EXECBIN}
68:         ldd ${EXECBIN}
69:         ident ${ALLSOURCES} ${OBJECTS} ${EXECBIN}
70:
71: again :
72:         ${GMAKE} spotless deps ci all lis
73:
74: ifeq (${NEEDINCL},)
75: include ${DEPSFILE}
76: endif
77:
```

```
 1: # Makefile.deps created Thu Feb 18 20:38:40 PST 2010
 2: comparex.o : comparex.cc
 3: comparex.o : comparex.ccti
 4: comparex.o : comparex.h
 5: comparex.o : trace.h
 6: listmap.o : comparex.h
 7: listmap.o : listmap.cc
 8: listmap.o : listmap.ccti
 9: listmap.o : listmap.h
10: listmap.o : pairx.h
11: listmap.o : trace.h
12: main.o : comparex.h
13: main.o : listmap.h
14: main.o : main.cc
15: main.o : pairx.h
16: main.o : trace.h
17: main.o : util.h
18: pairx.o : pairx.cc
19: pairx.o : pairx.ccti
20: pairx.o : pairx.h
21: pairx.o : trace.h
22: trace.o : trace.cc
23: trace.o : trace.h
24: util.o : trace.h
25: util.o : util.cc
26: util.o : util.ccti
27: util.o : util.h
```

```
    1: ldd keyvalue
    2:         libCstd.so.1 =>  /usr/lib/libCstd.so.1
    3:         libCrun.so.1 =>  /usr/lib/libCrun.so.1
    4:         libm.so.2 =>     /lib/libm.so.2
    5:         libc.so.1 =>     /lib/libc.so.1
    6: ident comparex.h listmap.h pairx.h trace.h util.h main.cc comparex.cc listmap.cc
 pairx.cc trace.cc util.cc comparex.ccti listmap.ccti pairx.ccti util.ccti Makefile REA
DME main.o comparex.o listmap.o pairx.o trace.o util.o keyvalue
    7: comparex.h:
    8:      $Id: comparex.h,v 1.5 2010-02-12 17:35:44-08 - - $
    9:
   10: listmap.h:
   11:      $Id: listmap.h,v 1.6 2010-02-18 16:13:44-08 - - $
   12:
   13: pairx.h:
   14:      $Id: pairx.h,v 1.4 2010-02-11 20:28:50-08 - - $
   15:
   16: trace.h:
   17:      $Compiled: " __FILE__ " " __DATE__ " " __TIME__ " $
   18:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
   19:
   20: util.h:
   21:      $Id: util.h,v 1.3 2010-02-12 17:35:44-08 - - $
   22:
   23: main.cc:
   24:      $Id: main.cc,v 1.6 2010-02-18 16:13:44-08 - - $
   25:
   26: comparex.cc:
   27:      $Id: comparex.cc,v 1.7 2010-02-18 20:36:31-08 - - $
   28:
   29: listmap.cc:
   30:      $Id: listmap.cc,v 1.10 2010-02-18 16:13:44-08 - - $
   31:
   32: pairx.cc:
   33:      $Id: pairx.cc,v 1.6 2010-02-12 15:03:23-08 - - $
   34:
   35: trace.cc:
   36:      $Id: trace.cc,v 1.1 2010-02-09 20:25:52-08 - - $
   37:
   38: util.cc:
   39:      $Id: util.cc,v 1.3 2010-02-12 15:03:23-08 - - $
   40:
   41: comparex.ccti:
   42:      $Id: comparex.ccti,v 1.2 2010-02-18 20:38:40-08 - - $
   43:
   44: listmap.ccti:
   45:      $Id: listmap.ccti,v 1.5 2010-02-18 20:38:40-08 - - $
   46:
   47: pairx.ccti:
   48:      $Id: pairx.ccti,v 1.3 2010-02-18 20:38:40-08 - - $
   49:
   50: util.ccti:
   51:      $Id: util.ccti,v 1.3 2010-02-18 20:38:40-08 - - $
   52:
   53: Makefile:
   54:      $Id: Makefile,v 1.7 2010-02-18 20:36:31-08 - - $
   55:
   56: README:
   57:
   58: main.o:
   59:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
   60:      $Id: comparex.h,v 1.5 2010-02-12 17:35:44-08 - - $
   61:      $Id: pairx.h,v 1.4 2010-02-11 20:28:50-08 - - $
   62:      $Id: listmap.h,v 1.6 2010-02-18 16:13:44-08 - - $
```

```
  63:      $Id: util.h,v 1.3 2010-02-12 17:35:44-08 - - $
  64:      $Id: main.cc,v 1.6 2010-02-18 16:13:44-08 - - $
  65:      $Compiled: main.cc Feb 18 2010 20:38:40 $
  66:
  67: comparex.o:
  68:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
  69:      $Id: comparex.h,v 1.5 2010-02-12 17:35:44-08 - - $
  70:      $Id: comparex.ccti,v 1.2 2010-02-18 20:38:40-08 - - $
  71:      $Compiled: comparex.ccti Feb 18 2010 20:38:41 $
  72:      $Id: comparex.cc,v 1.7 2010-02-18 20:36:31-08 - - $
  73:      $Compiled: comparex.cc Feb 18 2010 20:38:41 $
  74:
  75: listmap.o:
  76:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
  77:      $Id: comparex.h,v 1.5 2010-02-12 17:35:44-08 - - $
  78:      $Id: pairx.h,v 1.4 2010-02-11 20:28:50-08 - - $
  79:      $Id: listmap.h,v 1.6 2010-02-18 16:13:44-08 - - $
  80:      $Id: listmap.ccti,v 1.5 2010-02-18 20:38:40-08 - - $
  81:      $Compiled: listmap.ccti Feb 18 2010 20:38:41 $
  82:
  83: pairx.o:
  84:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
  85:      $Id: pairx.h,v 1.4 2010-02-11 20:28:50-08 - - $
  86:      $Id: pairx.ccti,v 1.3 2010-02-18 20:38:40-08 - - $
  87:      $Compiled: pairx.ccti Feb 18 2010 20:38:42 $
  88:      $Id: pairx.cc,v 1.6 2010-02-12 15:03:23-08 - - $
  89:      $Compiled: pairx.cc Feb 18 2010 20:38:42 $
  90:
  91: trace.o:
  92:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
  93:      $Id: trace.cc,v 1.1 2010-02-09 20:25:52-08 - - $
  94:      $Compiled: trace.cc Feb 18 2010 20:38:42 $
  95:
  96: util.o:
  97:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
  98:      $Id: util.h,v 1.3 2010-02-12 17:35:44-08 - - $
  99:      $Id: util.ccti,v 1.3 2010-02-18 20:38:40-08 - - $
 100:      $Compiled: util.ccti Feb 18 2010 20:38:42 $
 101:      $Id: util.cc,v 1.3 2010-02-12 15:03:23-08 - - $
 102:      $Compiled: util.cc Feb 18 2010 20:38:42 $
 103:
 104: keyvalue:
 105:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
 106:      $Id: comparex.h,v 1.5 2010-02-12 17:35:44-08 - - $
 107:      $Id: pairx.h,v 1.4 2010-02-11 20:28:50-08 - - $
 108:      $Id: listmap.h,v 1.6 2010-02-18 16:13:44-08 - - $
 109:      $Id: util.h,v 1.3 2010-02-12 17:35:44-08 - - $
 110:      $Id: main.cc,v 1.6 2010-02-18 16:13:44-08 - - $
 111:      $Compiled: main.cc Feb 18 2010 20:38:40 $
 112:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
 113:      $Id: comparex.h,v 1.5 2010-02-12 17:35:44-08 - - $
 114:      $Id: comparex.ccti,v 1.2 2010-02-18 20:38:40-08 - - $
 115:      $Compiled: comparex.ccti Feb 18 2010 20:38:41 $
 116:      $Id: comparex.cc,v 1.7 2010-02-18 20:36:31-08 - - $
 117:      $Compiled: comparex.cc Feb 18 2010 20:38:41 $
 118:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
 119:      $Id: comparex.h,v 1.5 2010-02-12 17:35:44-08 - - $
 120:      $Id: pairx.h,v 1.4 2010-02-11 20:28:50-08 - - $
 121:      $Id: listmap.h,v 1.6 2010-02-18 16:13:44-08 - - $
 122:      $Id: listmap.ccti,v 1.5 2010-02-18 20:38:40-08 - - $
 123:      $Compiled: listmap.ccti Feb 18 2010 20:38:41 $
 124:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
 125:      $Id: pairx.h,v 1.4 2010-02-11 20:28:50-08 - - $
 126:      $Id: pairx.ccti,v 1.3 2010-02-18 20:38:40-08 - - $
```

```
127:      $Compiled: pairx.ccti Feb 18 2010 20:38:42 $
128:      $Id: pairx.cc,v 1.6 2010-02-12 15:03:23-08 - - $
129:      $Compiled: pairx.cc Feb 18 2010 20:38:42 $
130:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
131:      $Id: trace.cc,v 1.1 2010-02-09 20:25:52-08 - - $
132:      $Compiled: trace.cc Feb 18 2010 20:38:42 $
133:      $Id: trace.h,v 1.3 2010-02-11 20:28:50-08 - - $
134:      $Id: util.h,v 1.3 2010-02-12 17:35:44-08 - - $
135:      $Id: util.ccti,v 1.3 2010-02-18 20:38:40-08 - - $
136:      $Compiled: util.ccti Feb 18 2010 20:38:42 $
137:      $Id: util.cc,v 1.3 2010-02-12 15:03:23-08 - - $
138:      $Compiled: util.cc Feb 18 2010 20:38:42 $
```