```
 1: // $Id: oclib.oh,v 1.24 2011-11-18 17:34:29-08 - - $
 2:
 3: #ifndef __OCLIB_OH__
 4: #define __OCLIB_OH__
 5:
 6: #ifdef __OCLIB_C__
 7: #   define __(ID)         __##ID
 8: #   define NONE__         void
 9: #   define VOID__(ID)     void __##ID
10: #   define BOOL__(ID)     ubyte __##ID
11: #   define CHAR__(ID)     ubyte __##ID
12: #   define INT__(ID)      int __##ID
13: #   define STRING__(ID)   ubyte *__##ID
14: #   define STRINGS__(ID)  ubyte **__##ID
15: #   define null           0
16: #   define false          0
17: #   define true           1
18: typedef unsigned char ubyte;
19: void *xcalloc (int nelem, int size);
20: #else
21: #   define EOF            (-1)
22: #   define __(ID)         ID
23: #   define NONE__
24: #   define VOID__(ID)     void ID
25: #   define BOOL__(ID)     bool ID
26: #   define CHAR__(ID)     char ID
27: #   define INT__(ID)      int ID
28: #   define STRING__(ID)   string ID
29: #   define STRINGS__(ID)  string[] ID
30: VOID__(__assert_fail) (STRING__(expr), STRING__(file), INT__(line));
31: #endif
32:
33: VOID__(putb) (BOOL__(b));
34: VOID__(putc) (CHAR__(c));
35: VOID__(puti) (INT__(i));
36: VOID__(puts) (STRING__(s));
37: VOID__(endl) (NONE__);
38: INT__(getc) (NONE__);
39: STRING__(getw) (NONE__);
40: STRING__(getln) (NONE__);
41: STRINGS__ (getargv) (NONE__);
42: VOID__(exit) (int status);
43: #define assert(expr) \
44:         {if (! (expr)) __(__assert_fail) (#expr, __FILE__, __LINE__);}
45:
46: #endif
47:
```

```
1: // $Id: 00-trivial.oc,v 1.1 2011-09-15 18:50:16-07 - - $
2: //
3: // This program does nothing then returns an exit status of 0.
4: //
```

```
1: // $Id: 01-hello.oc,v 1.1 2011-09-15 18:50:16-07 - - $
2: // Simple hello world program.
3:
4: #include "oclib.oh"
5:
6: puts ("Hello, world!\n");
```

```
1: // $Id: 03-test3.oc,v 1.1 2011-09-15 18:50:16-07 - - $
2:
3: int a = 3;
4: int b = 8;
5: int c = a + b;
6: a = b + c;
7: puti (a);
8: putc ('\n');
9:
```

```
 1: // $Id: 04-test4.oc,v 1.1 2011-09-15 18:50:16-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: struct foo {
 6:    int a;
 7: }
 8:
 9: int a = 6;
10: foo b = new foo ();
11: b.a = 8;
12: a = a * b.a + 6;;
13:
14: puti (a);
15: putc (' ');
16: puti (b.a);
17: endl ();
18:
```

```
 1: // $Id: 05-test5.oc,v 1.1 2011-09-15 18:50:16-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: bool a = true;
 6: a = false;
 7:
 8: // The following will scan and parse, but error out at type check.
 9: a = b + c * d;
10:
```

```
 1: // $Id: 06-test6.oc,v 1.2 2011-09-19 14:25:40-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: int f0 ();
 6: int f1 (int a);
 7: int f2 (int a, int b);
 8: int f3 (string a, string b, string c);
 9: int f4 (foo a, bar b);
10: string s = "a";
11: string[] sa = new string[10];
12:
```

```
 1: // $Id: 10-hundred.oc,v 1.1 2011-09-15 18:50:16-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: int count = 0;
 6: while (count <= 100) {
 7:    count = count + 1;
 8:    puti (count);
 9:    endl ();
10: }
```

```
 1: // $Id: 11-numbers.oc,v 1.1 2011-09-15 18:50:16-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: int number = 1;
 6: while (number > 0) {
 7:    puti (number);
 8:    putc ('\n');
 9:    number = number + number;
10: }
11: puti (number);
12: putc ('\n');
13:
```

```
1: // $Id: 12-elseif.oc,v 1.1 2011-09-15 18:50:16-07 - - $
2:
3: int a = 3;
4:
5: if (a == 1) puts ("one");
6: else if (a == 2) puts ("two");
7: else if (a == 3) puts ("three");
8: else puts ("many");
9: endl ();
```

```
 1: // $Id: 13-assertfail.oc,v 1.3 2011-11-08 14:53:37-08 - - $
 2:
 3: #undef __OCLIB_OH__
 4: #include "oclib.oh"
 5:
 6: puts (getargv()[0]);
 7: puts (" was compiled ");
 8: puts (__DATE__);
 9: puts (" @ ");
10: puts (__TIME__);
11: endl ();
12: assert ("assert" == "fail");
13:
```

```
 1: // $Id: 14-ocecho.oc,v 1.2 2011-11-16 23:08:30-08 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: string[] argv = getargv ();
 6: int argi = 1;
 7: while (argv[argi] != null) {
 8:    if (argi > 1) putc (' ');
 9:    puts (argv[argi]);
10:    argi = argi + 1;
11: }
12: endl ();
13:
```

```
 1: // $Id: 20-fib-array.oc,v 1.2 2011-09-20 19:23:27-07 - - $
 2: //
 3: // Put Fibonacci numbers in an array, then print them.
 4: //
 5:
 6: #define FIB_SIZE 30
 7: int[] fibonacci = new int[FIB_SIZE];
 8:
 9: fibonacci[0] = 0;
10: fibonacci[1] = 1;
11:
12: int index = 2;
13: while (index < FIB_SIZE) {
14:    fibonacci[index] = fibonacci[index - 1] + fibonacci[index - 2];
15:    index = index + 1;
16: }
17:
18: index = 0;
19: puts ("Numeri di figlio Bonacci\n");
20: while (index < FIB_SIZE) {
21:    puts ("fibonacci[");
22:    puti (index);
23:    puts (" = ");
24:    puti (fibonacci[index]);
25:    endl ();
26:    index = index + 1;
27: }
```

```
 1: // $Id: 21-eratosthenes.oc,v 1.2 2011-09-19 14:25:40-07 - - $
 2:
 3: #include "oclib.oh"
 4: #define SIZE 100
 5: #define LOWPRIME 2
 6:
 7: bool[] sieve = new bool[SIZE];
 8: int index = LOWPRIME;
 9:
10: while (index < SIZE) {
11:    sieve[index] = true;
12:    index = index + 1;
13: }
14:
15: int prime = LOWPRIME;
16: while (prime < SIZE) {
17:    if (sieve[prime]) {
18:       index = prime * 2;
19:       while (index < SIZE) {
20:          sieve[index] = false;
21:          index = index + prime;
22:       }
23:    }
24:    prime = prime + 1;
25: }
26:
27: index = LOWPRIME;
28: while (index < SIZE) {
29:    if (sieve[index]) {
30:       puti (index);
31:       endl ();
32:    }
33: }
34:
```

```
 1: // $Id: 23-atoi.oc,v 1.6 2011-10-28 18:23:43-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: int atoi (string str) {
 6:    assert (str != null);
 7:    bool neg = false;
 8:    int num = 0;
 9:    int digit = 0;
10:    if (str[0] != '\0') {
11:       if (str[0] == '-') {
12:          digit = digit + 1;
13:          neg = true;
14:       }
15:       bool contin = true;
16:       while (contin) {
17:          if (str[digit] == '\0') {
18:             contin = false;
19:          }else {
20:             char c = str[digit];
21:             digit = digit + 1;
22:             if (c < '0') contin = false;
23:             else if (c > '9') contin = false;
24:             else num = num * 10 + ord c - ord '0';
25:          }
26:       }
27:       if (neg) num = - num;
28:    }
29:    return num;
30: }
31:
32: int argi = 0;
33: string[] argv = getargv ();
34: while (argv[argi] != null) {
35:    string arg = argv[argi];
36:    puts (arg);
37:    puts (" = ");
38:    puts (atoi (arg));
39:    endl ();
40: }
41:
```

```
 1: // $Id: 30-fac-fnloop.oc,v 1.4 2011-11-16 20:18:50-08 - - $
 2: //
 3: // Function uses a loop to compute factorial.
 4: //
 5:
 6: #include "oclib.oh"
 7:
 8: int fac (int n) {
 9:    int f = 1;
10:    while (n > 1) {
11:       f = f * n;
12:       n = n - 1;
13:    }
14:    return f;
15: }
16:
17: int n = 1;
18: while (n <= 5) {
19:    puti (fac (n));
20:    endl ();
21:    n = n + 1;
22: }
23:
```

```
 1: // $Id: 31-fib-2supn.oc,v 1.1 2011-10-20 21:31:44-07 - - $
 2: //
 3: // Very slow program, computes Fibonacci numbers with O(2^n) speed.
 4: //
 5:
 6: #include "oclib.oh"
 7:
 8: int fibonacci (int n) {
 9:    if (n < 2) return n;
10:    return fibonacci (n - 1) + fibonacci (n - 2);
11: }
12:
13: // Main program.
14:
15: int n = 0;
16: while (n < 10) {
17:    puts ("fibonacci(");
18:    puti (n);
19:    puts (" = ");
20:    puti (fibonacci (n));
21:    endl ();
22: }
23:
```

```
 1: // $Id: 40-arraystack.oc,v 1.4 2011-10-28 18:24:19-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: #define EMPTY (-1)
 6: struct stack {
 7:    string[] data;
 8:    int size;
 9:    int top;
10: }
11:
12: stack new_stack (int size) {
13:    stack stack = new stack (); // Zeros out both fields.
14:    stack.data = new string[size]; // Array of null pointers.
15:    stack.size = size;
16:    stack.top = EMPTY;
17:    return stack;
18: }
19:
20: void push (stack stack, string str) {
21:    assert (stack.top < stack.size - 1);
22:    stack.top = stack.top + 1;
23:    stack.data[stack.top] = str;
24: }
25:
26: string pop (stack stack) {
27:    assert (stack.top != EMPTY);
28:    string tmp = stack.data[top];
29:    top = top - 1;
30:    return tmp;
31: }
32:
33: bool empty (stack stack) {
34:    return stack.top == EMPTY;
35: }
36:
37: // Main program.
38: string[] argv = getargv ();
39: stack stack = new_stack ();
40:
41: int argi = 0;
42: while (argv[argi] != null) {
43:    push (stack, argv[argi]);
44:    argi = argi + 1;
45: }
46:
47: while (! empty (stack)) {
48:    puts (pop (stack));
49:    endl ();
50: }
51:
```

```
 1: // $Id: 41-linkedstack.oc,v 1.4 2011-10-28 18:24:19-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: struct stack {
 6:    node top;
 7: }
 8:
 9: struct node {
10:    string data;
11:    node link;
12: }
13:
14: stack new_stack (int size) {
15:    stack stack = new stack ();
16:    stack.top = null;
17:    return stack;
18: }
19:
20: void push (stack stack, string str) {
21:    assert (stack != null);
22:    node tmp = new node ();
23:    tmp.data = str;
24:    tmp.link = stack.top;
25:    stack.top = tmp;
26: }
27:
28: string pop (stack stack) {
29:    assert (stack != null);
30:    assert (! empty (stack));
31:    string tmp = stack.top.data;
32:    stack.top = stack.top.link;
33:    return tmp;
34: }
35:
36: bool empty (stack stack) {
37:    assert (stack != null);
38:    return stack.top == null;
39: }
40:
41: // Main program.
42:
43: string[] argv = getargv ();
44: stack stack = new_stack ();
45: int argi = 0;
46:
47: while (argv[argi] != null) {
48:    push (stack, argv[argi]);
49:    argi = argi + 1;
50: }
51:
52: while (! empty (stack)) {
53:    puts (pop (stack));
54:    endl ();
55: }
56:
```

```
 1: // $Id: 42-viiiqueens.oc,v 1.3 2011-10-25 18:21:10-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: #define BOARD_SIZE 8
 6: int[] board = new int[BOARD_SIZE];
 7:
 8: bool is_safe (int newcol) {
 9:    int col = 0;
10:    while (col < newcol) {
11:       if (board[col] == board[newcol]) return false;
12:       int diagonal = board[col] - board[newcol];
13:       if (diagonal == col - newcol) return false;
14:       if (diagonal == newcol - col) return false;
15:       col = col + 1;
16:    }
17:    return true;
18: }
19:
20: void printqueens () {
21:    int col = 0;
22:    while (col < BOARD_SIZE) {
23:       putchar (chr (board[col] + ord '1'));
24:       col = col + 1;
25:    }
26:    putchar ('\n');
27: }
28:
29: void queens (int newcol) {
30:    if (newcol == BOARD_SIZE) printqueens ();
31:    else {
32:       int row = 0;
33:       while (row < BOARD_SIZE) {
34:          board[newcol] = row;
35:          if (is_safe (newcol)) queens (newcol + 1);
36:          row = row + 1;
37:       }
38:    }
39: }
40:
41: // Main program.
42: queens (0);
43:
```

```
 1: // $Id: 44-dot-product.oc,v 1.2 2011-09-20 19:23:27-07 - - $
 2:
 3: int dot_product (int size, int[] vec1, int[] vec2) {
 4:    int index = 0;
 5:    int dot = 0;
 6:    while (index < size) {
 7:       dot = dot + vec1[index] * vec2[index];
 8:       index = index + 1;
 9:    }
10:    return dot;
11: }
12:
13: #define SIZE 10
14: int[] vec1 = new int[SIZE];
15: int[] vec2 = new int[SIZE];
16: int i = 0;
17: while (i < SIZE) {
18:    vec1[i] = i + 10;
19:    vec2[i] = i * 10;
20: }
21: puti (dot_product (SIZE, vec1, fec2));
22: endl ();
```

```
 1: // $Id: 45-towers-of-hanoi.oc,v 1.2 2011-09-20 19:44:21-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: void move (string src, string dst) {
 6:    puts ("Move a disk from ");
 7:    puts (src);
 8:    puts (" to ");
 9:    puts (dst);
10:    puts (".\n");
11: }
12:
13: void towers (int ndisks, string src, string tmp, string dst) {
14:    if (ndisks < 1) return;
15:    towers (ndisks - 1, src, tmp);
16:    move (src, dst);
17:    towers (ndisks - 1, tmp, dst);
18: }
19:
20: towers (4, "Source", "Temporary", "Destination");
21:
```

```
 1: // $Id: 53-insertionsort.oc,v 1.3 2011-10-28 14:32:36-07 - - $
 2: //
 3: // Use insertion sort to print argv in osrted order.
 4: //
 5:
 6: #include "oclib.oh"
 7:
 8: int strcmp (string s1, string s2) {
 9:    int index = 0;
10:    bool contin = true;
11:    while (contin) {
12:       char s1c = s1[index];
13:       char s2c = s2[index];
14:       int cmp = ord s1c - ord s2c;
15:       if (cmp != 0) return cmp;
16:       if (s1c == '\0') contin = false;
17:       index = index + 1;
18:    }
19:    return 0;
20: }
21:
22: void insertion_sort (int size, string[] array) {
23:    int sorted = 1;
24:    while (sorted < size) {
25:       int slot = sorted;
26:       string element = array[slot];
27:       bool contin = true;
28:       while (contin) {
29:          if (slot == 0) {
30:             contin = false;
31:          }else if (strcmp (array[slot - 1], element) <= 0) {
32:             contin = false;
33:          }else {
34:             array[slot] = array[slot - 1];
35:             slot = slot - 1;
36:          }
37:       }
38:       array[slot] = element;
39:       sorted = sorted + 1;
40:    }
41: }
42:
43: void print_array (string label, int size, string[] array) {
44:    endl ();
45:    puts (label);
46:    puts (":\n");
47:    int index = 0;
48:    while (index < size) {
49:       puts (array[index]);
50:       endl ();
51:       index = index + 1;
52:    }
53: }
54:
55: string[] argv = getargv ();
56: int argc = 0;
57: while (argv[argc] != null) argc = argc + 1;
58: print_argv ("unsorted", argc, argv);
59: insertion_sort (argc, argv);
60: print_argv ("sorted", argc, argv);
61:
```

```
1: char O[9];Q(l,b,d){int o=8,p=1,q=1<<
2: l|1<<22-l;for(;l>7?!write(1,O,9):o--
3: ;)O[l]=56-o,b&p|d&q||Q(l+1,b|p,d|q),
4: p*=2,q*=2;}main(){O[8]=10;Q(0,0,0);}
```

```
 1: // $Id: 91-typecheck.oc,v 1.1 2011-11-07 12:09:34-08 - - $
 2: //
 3: // This file should scan and parse correctly,
 4: // but fail to type check.
 5: //
 6:
 7: int[] a = null;
 8: reference[] a = new string[10];
 9: void foo ();
10: void foo (int a);
11: void foo (int[] a, int[] b) {int x = a + b;}
12: struct foo { int a; int b; }
13:
14: a + b;
15: f ();
16: f (x, y+3, z);
17: foo + bar;
18: a = b = c = d;
19: test = abc + def + ghi;
20: this + 23 * a + "hello";
21: while (a < b) f = f + 1;
22: return 3 + 4;
23: a[i] = b[j];
24: return;
25: while (true) {a = 3; b = 4; }
26: if (a == b) f (x);
27: if (a != b) y = 3; else f (y, z);
28:
```

```
1: /*
2: This is an unterminated comment.
3: It would cause cpp to error out.
4: When cpp returns a non-zero exit code,
5: so should your compiler.
6: $Id: 92-uncomment.oc,v 1.1 2011-09-15 18:50:16-07 - - $
7:
8: int main (int argc, char **argv) {
```

```
 1: // $Id: 93-semantics.oc,v 1.1 2011-11-01 22:02:19-07 - - $
 2: // This code should scan and parse correctly,
 3: // but fail to type check.
 4: int[] a = null;
 5: int[] b = null;
 6: int c = a + b; // can’t add arrays
 7: void[] f() {}; // can’t hae void[]
 8: void n = null; // can’t have void vars
 9: bool x = a < b; // can’t compare pointers <
10: bool y = a==b; // this is ok
```

```
1: // $Id: 94-syntax.oc,v 1.1 2011-11-01 22:02:19-07 - - $
2: k
3: int f() {
4: int a = ;
5: return foo;
6: public static void main (String[] args) {
7:    System.exit (255);
8: }
9:
```

```
 1: // $Id: 95-cobol.oc,v 1.1 2011-11-01 22:02:19-07 - - $
 2:
 3: 000100 IDENTIFICATION DIVISION.
 4: 000200 PROGRAM-ID.     HELLOWORLD.
 5: 000300
 6: 000400*
 7: 000500 ENVIRONMENT DIVISION.
 8: 000600 CONFIGURATION SECTION.
 9: 000700 SOURCE-COMPUTER. RM-COBOL.
10: 000800 OBJECT-COMPUTER. RM-COBOL.
11: 000900
12: 001000 DATA DIVISION.
13: 001100 FILE SECTION.
14: 001200
15: 100000 PROCEDURE DIVISION.
16: 100100
17: 100200 MAIN-LOGIC SECTION.
18: 100300 BEGIN.
19: 100400     DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
20: 100500     DISPLAY "Hello world!" LINE 15 POSITION 10.
21: 100600     STOP RUN.
22: 100700 MAIN-LOGIC-EXIT.
23: 100800     EXIT.
```