```
// $Id: 00-trivial.oc,v 1.1 2019-04-16 12:14:45-07 - - $

int main(){}
```

```
// $Id: 01-hello.oc,v 1.1 2019-04-16 12:14:45-07 - - $
// Simple hello world program.

#include "oclib.h"

int main() {
   putstr ("Hello, world!\n");
   return SUCCESS;
}

```

```
// $Id: 03-test3.oc,v 1.1 2019-04-16 12:14:45-07 - - $

#include "oclib.h"

int main() {
   int a = 3;
   int b = 8;
   int c = a + b;
   a = b + c;
   putint (a);
   putchr ('\n');
   return SUCCESS;
}

```

```
// $Id: 04-test4.oc,v 1.1 2019-04-16 12:14:45-07 - - $

#include "oclib.h"

struct foo {
   int a;
};

int main() {
   int a = 6;
   ptr<struct foo> b = alloc<struct foo>();
   b->a = 8;
   a = a * b->a + 6;;
   putint (a);
   putchr (' ');
   putint (b->a);
   putchr ('\n');
   return SUCCESS;
}

```

```
// $Id: 06-test6.oc,v 1.1 2019-04-16 12:14:45-07 - - $

#include "oclib.h"

struct foo {};
struct bar {};

int f0();
int f1 (int a);
int f2 (int a, int b);
int f3 (string a, string b, string c);
int f4 (ptr<struct foo> a, ptr<struct bar> b);

int main() {
   string s = "a";
   array<string> sa = alloc<array<string>>(10);
   return SUCCESS;
}

```

```
// $Id: 07-assert.oc,v 1.2 2019-04-16 13:14:01-07 - - $

#include "oclib.h"

int main() {
   assert ("nullptr" == nullptr);
   return FAILURE;
}

```

```
// $Id: 10-hundred.oc,v 1.2 2019-04-17 13:23:14-07 - - $

#include "oclib.h"

int main() {
   int count = 0;
   while (count < 16) {
      count = count + 1;
      putint (count);
      putchr ('\n');
   }
   return SUCCESS;
}

```

```
// $Id: 11-numbers.oc,v 1.2 2019-04-18 14:06:21-07 - - $

#include "oclib.h"

int main() {
   int number = 1;
   BOOL done = FALSE;
   while (not done) {
      putint (number);
      putchr ('\n');
      if (number <= 0) done = TRUE;
      number = number + number;
   }
   return SUCCESS;
}

```

```
// $Id: 12-elseif.oc,v 1.1 2019-04-16 12:14:45-07 - - $

#include "oclib.h"

int main (int argc) {
   if (argc == 1) putstr ("one");
   else if (argc == 2) putstr ("two");
   else if (argc == 3) putstr ("three");
   else putstr ("many");
   putchr ('\n');
   return SUCCESS;
}

```

```
// $Id: 13-macros.oc,v 1.3 2019-04-23 15:25:57-07 - - $

#include "oclib.h"

int main (int argc, array<string> argv) {
   putstr (argv[0]);
   putstr (": ");
   putstr (__FILE__);
   putchr ('[');
   putint (__LINE__);
   putstr ("] compiled ");
   putstr (__DATE__);
   putstr (" ");
   putstr (__TIME__);
   putstr ("\n");
   return SUCCESS;
}

```

```
// $Id: 14-ocecho.oc,v 1.1 2019-04-16 12:14:45-07 - - $

#include "oclib.h"

int main(int argc, array<string> argv) {
   int argi = 1;
   while (argi < argc) {
      if (argi > 1) putchr (' ');
      putstr (argv[argi]);
      argi = argi + 1;
   }
   putstr ("\n");
   return SUCCESS;
}

```

```
// $Id: 20-fib-array.oc,v 1.3 2019-04-23 15:22:03-07 - - $
//
// Put Fibonacci numbers in an array, then print them.
//

#include "oclib.h"

#define FIB_SIZE 30

int main() {
   array<int> fibonacci = alloc<array<int>> (FIB_SIZE);
   fibonacci[0] = 0;
   fibonacci[1] = 1;
   int index = 2;
   while (index < FIB_SIZE) {
      fibonacci[index] = fibonacci[index - 1] + fibonacci[index - 2];
      index = index + 1;
   }
   index = 0;
   putstr ("Numeri di figlio Bonacci\n");
   while (index < FIB_SIZE) {
      putstr ("fibonacci[");
      putint (index);
      putstr ("] = ");
      putint (fibonacci[index]);
      putchr ('\n');
      index = index + 1;
   }
   return SUCCESS;
}

```

```
// $Id: 21-eratosthenes.oc,v 1.1 2019-04-16 12:14:45-07 - - $

#include "oclib.h"

#define SIZE 100
#define LOWPRIME 2

int main() {
   int prime = LOWPRIME;
   int index = LOWPRIME;
   array<int> sieve = alloc<array<int>>(SIZE);

   while (index < SIZE) {
      sieve[index] = TRUE;
      index = index + 1;
   }

   while (prime < SIZE) {
      if (sieve[prime]) {
         index = prime * 2;
         while (index < SIZE) {
            sieve[index] = FALSE;
            index = index + prime;
         }
      }
      prime = prime + 1;
   }

   index = LOWPRIME;
   while (index < SIZE) {
      if (sieve[index]) {
         putint (index);
         putchr ('\n');
      }
      index = index + 1;
   }

   return SUCCESS;
}

```

```
// $Id: 23-atoi.oc,v 1.3 2019-04-23 15:22:03-07 - - $

#include "oclib.h"

int atoi (string str) {
   BOOL neg = FALSE;
   int num = 0;
   int digit = 0;
   assert (str != nullptr);
   if (str[0] != '\0') {
      if (str[0] == '-') {
         digit = digit + 1;
         neg = TRUE;
      }
      BOOL contin = TRUE;
      while (contin) {
         if (str[digit] == '\0') {
            contin = FALSE;
         }else {
            int chr = str[digit];
            digit = digit + 1;
            if (chr < '0') contin = FALSE;
            else if (chr > '9') contin = FALSE;
            else num = num * 10 + chr - '0';
         }
      }
      if (neg) num = - num;
   }
   return num;
}

int main (int argc, array<string> argv) {
   int argi = 1;
   string arg = nullptr;
   while (argi < argc) {
      arg = argv[argi];
      putstr (arg);
      putstr (" = ");
      putint (atoi (arg));
      putchr ('\n');
      argi = argi + 1;
   }
   return SUCCESS;
}

```

```
// $Id: 30-fac-fnloop.oc,v 1.1 2019-04-16 12:14:45-07 - - $
//
// Function uses a loop to compute factorial.
//

#include "oclib.h"

int fac (int n) {
   int f = 1;
   while (n > 1) {
      f = f * n;
      n = n - 1;
   }
   return f;
}

int main() {
   int n = 1;
   while (n <= 5) {
      putint (fac (n));
      putchr ('\n');
      n = n + 1;
   }
   return SUCCESS;
}

```

```
// $Id: 31-fib-2supn.oc,v 1.1 2019-04-16 12:14:45-07 - - $
//
// Very slow program, computes Fibonacci numbers with O(2^n) speed.
//

#include "oclib.h"

int fibonacci (int n) {
   if (n < 2) return n;
   return fibonacci (n - 1) + fibonacci (n - 2);
}

int main() {
   int n = 0;
   while (n < 10) {
      putstr ("fibonacci(");
      putint (n);
      putstr (") = ");
      putint (fibonacci (n));
      putchr ('\n');
      n = n + 1;
   }
   return SUCCESS;
}

```

```
// $Id: 33-collatz.oc,v 1.4 2019-05-08 15:20:01-07 - - $

//
// Compute the number of iterations needed for the Collatz conjecture.
//

#include "oclib.h"

int collatz (int n) {
   int c = 0;
   while (n != 1) {
      c = c + 1;
      if (n % 2 == 1) n = 3 * n + 1;
                 else n = n / 2;
   }
   return c;
}

void test (int n) {
   int c = collatz (n);
   putstr ("collatz(");
   putint (n);
   putstr (") = ");
   putint (c);
   putchr ('\n');
}

int main() {
   test (3);
   test (50);
   return SUCCESS;
}

```

```
// $Id: 40-arraystack.oc,v 1.2 2019-04-17 13:23:14-07 - - $

#include "oclib.h"

#define EMPTY (-1)

struct stack {
   array<string> data;
   int size;
   int top;
};

ptr<struct stack> new_stack (int size) {
   ptr<struct stack> stack = alloc<struct stack>();
   stack->data = alloc<array<string>> (size);
   stack->size = size;
   stack->top = EMPTY;
   return stack;
}

void push (ptr<struct stack> stack, string str) {
   assert (stack->top < stack->size - 1);
   stack->top = stack->top + 1;
   stack->data[stack->top] = str;
}

string pop (ptr<struct stack> stack) {
   string tmp = stack->data[stack->top];
   assert (stack->top != EMPTY);
   stack->top = stack->top - 1;
   return tmp;
}

int empty (ptr<struct stack> stack) {
   return stack->top == EMPTY;
}

int main (int argc, array<string> argv) {
   ptr<struct stack> stack = new_stack (100);
   int argi = 0;
   while (argi < argc) {
      push (stack, argv[argi]);
      argi = argi + 1;
   }
   while (not empty (stack)) {
      putstr (pop (stack));
      putchr ('\n');
   }
   return SUCCESS;
}

```

```
// $Id: 41-linkedstack.oc,v 1.4 2019-05-06 12:54:55-07 - - $

#include "oclib.h"

struct node {
   string data;
   ptr <struct node> link;
};

struct stack {
   ptr <struct node> top;
};

int empty (ptr<struct stack> stack) {
   assert (stack != nullptr);
   return stack->top == nullptr;
}

ptr<struct stack> new_stack() {
   ptr<struct stack> stack = alloc<struct stack>();
   stack->top = nullptr;
   return stack;
}

void push (ptr<struct stack> stack, string str) {
   ptr <struct node> tmp = alloc<struct node>();
   assert (stack != nullptr);
   tmp->data = str;
   tmp->link = stack->top;
   stack->top = tmp;
}

string pop (ptr<struct stack> stack) {
   string tmp = stack->top->data;
   assert (stack != nullptr);
   assert (not empty (stack));
   stack->top = stack->top->link;
   return tmp;
}

int main (int argc, array<string> argv) {
   int argi = 0;
   ptr<struct stack> stack = new_stack();
   while (argi < argc) {
      push (stack, argv[argi]);
      argi = argi + 1;
   }
   while (not empty (stack)) {
      putstr (pop (stack));
      putchr ('\n');
   }
   return SUCCESS;
}

```

```
// $Id: 42-viiiqueens.oc,v 1.1 2019-04-16 12:14:45-07 - - $

#include "oclib.h"

#define BOARD_SIZE 8
array<int> board = nullptr;

int is_safe (int newcol) {
   int col = 0;
   int diagonal = 0;
   while (col < newcol) {
      if (board[col] == board[newcol]) return FALSE;
      diagonal = board[col] - board[newcol];
      if (diagonal == col - newcol) return FALSE;
      if (diagonal == newcol - col) return FALSE;
      col = col + 1;
   }
   return TRUE;
}

void printqueens() {
   int col = 0;
   while (col < BOARD_SIZE) {
      putchr (board[col] + '1');
      col = col + 1;
   }
   putchr ('\n');
}

void queens (int newcol) {
   int row = 0;
   if (newcol == BOARD_SIZE) printqueens();
   else {
      while (row < BOARD_SIZE) {
         board[newcol] = row;
         if (is_safe (newcol)) queens (newcol + 1);
         row = row + 1;
      }
   }
}

int main() {
   board = alloc<array<int>> (BOARD_SIZE);
   queens (0);
   return SUCCESS;
}

```

```
// $Id: 44-dot-product.oc,v 1.2 2019-04-23 15:22:03-07 - - $

#include "oclib.h"

int dot_product (int size, array<int> vec1, array<int> vec2) {
   int index = 0;
   int dot = 0;
   while (index < size) {
      dot = dot + vec1[index] * vec2[index];
      index = index + 1;
   }
   return dot;
}

#define SIZE 10

int main() {
   array<int> vec1 = alloc<array<int>> (SIZE);
   array<int> vec2 = alloc<array<int>> (SIZE);
   int index = 0;
   while (index < SIZE) {
      vec1[index] = index + 10;
      vec2[index] = index * 10;
      index = index + 1;
   }
   putint (dot_product (SIZE, vec1, vec2));
   putchr ('\n');
   return SUCCESS;
}

```

```
// $Id: 45-towers-of-hanoi.oc,v 1.2 2019-04-25 12:55:47-07 - - $

#include "oclib.h"

void move (string src, string dst) {
   putstr ("Move a disk from the ");
   putstr (src);
   putstr (" to the ");
   putstr (dst);
   putstr (".\n");
}

void towers (int ndisks, string src, string tmp, string dst) {
   if (ndisks < 1) return;
   towers (ndisks - 1, src, dst, tmp);
   move (src, dst);
   towers (ndisks - 1, tmp, src, dst);
}

int main (int argc, array<string> argv) {
   assert (argc == 2);
   int count = argv[1][0] - '0';
   assert (count > 0);
   assert (count < 9);
   putstr (argv[0]);
   putstr (": ");
   putint (count);
   putstr (" disks\n");
   towers (count, "source", "temporary", "distination");
   return SUCCESS;
}

```

```
// $Id: 51-stringcat.oc,v 1.3 2019-04-23 15:22:03-07 - - $
//
// Allocate and concatenate strings.
//

#include "oclib.h"

int strlen (string str) {
   int len = 0;
   while (str[len] != '\0') len = len + 1;
   return len;
}

void strcat (string dest, string src) {
   int pos = strlen (dest);
   int srcix = 0;
   while (src[srcix] != '\0') {
      dest[pos] = src[srcix];
      pos = pos + 1;
      srcix = srcix + 1;
   }
   dest[pos] = '\0';
}

int main (int argc, array<string> argv) {
   int length = 0;
   int argi = 1;
   while (argi < argc) {
      length = length + strlen (argv[argi]) + 2;
      argi = argi + 1;
   }
   putstr ("concat length = ");
   putint (length);
   putchr ('\n');
   string concat = alloc<string> (length);
   argi = 1;
   putchr ('[');
   while (argi < argc) {
      strcat (concat, "(");
      strcat (concat, argv[argi]);
      strcat (concat, ")");
      argi = argi + 1;
   }
   putstr (concat);
   putstr ("]\n");
   return SUCCESS;
}
```

```
// $Id: 53-insertionsort.oc,v 1.5 2019-05-06 12:15:43-07 - - $
//
// Use insertion sort to print argv in sorted order.
//

#include "oclib.h"

int strcmp (string s1, string s2) {
   int index = 0;
   BOOL contin = TRUE;
   int s1c = 0;
   int s2c = 0;
   int cmp = 0;
   while (contin) {
      s1c = s1[index];
      s2c = s2[index];
      cmp = s1c - s2c;
      if (cmp != 0) return cmp;
      if (s1c == '\0') contin = FALSE;
      if (s2c == '\0') contin = FALSE;
      index = index + 1;
   }
   return 0;
}

void insertion_sort (int size, array<string> words) {
   int sorted = 1;
   int slot = 0;
   string element = nullptr;
   BOOL contin = FALSE;
   while (sorted < size) {
      slot = sorted;
      element = words[slot];
      contin = TRUE;
      while (contin) {
         if (slot == 0) {
            contin = FALSE;
         }else if (strcmp (words[slot - 1], element) <= 0) {
            contin = FALSE;
         }else {
            words[slot] = words[slot - 1];
            slot = slot - 1;
         }
      }
      words[slot] = element;
      sorted = sorted + 1;
   }
}

```

```

void print_array (int size, array<string> words) {
   int index = 0;
   while (index < size) {
      putstr (words[index]);
      putchr ('\n');
      index = index + 1;
   }
}

int read_words (int size, array<string> words) {
   int count = 0;
   string word = nullptr;
   while (TRUE) {
      if (count == size) return count;
      word = getstr();
      if (word == nullptr) return count;
      words[count] = word;
      count = count + 1;
   }
}

int main() {
   int count = 100;
   array<string> words = alloc<array<string>>(count);
   count = read_words (count, words);
   insertion_sort (count, words);
   print_array (count, words);
   return SUCCESS;
}

```

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

```
// $Id: 91-typecheck.oc,v 1.1 2019-05-09 14:45:22-07 - - $
//
// This file should scan and parse correctly,
// but fail to type check, except for the global
// new string, which might be a syntax error
// or a semantic error.
//

int[] a = null;
reference[] a = new string[10];
void foo();
void foo (int a);
void foo (int[] a, int[] b) {int x = a + b;}
struct foo { int a; int b; }

int main() {
   a + b;
   f();
   f (x, y+3, z);
   foo + bar;
   a = b = c = d;
   test = abc + def + ghi;
   this + 23 * a + "hello";
   while (a < b) f = f + 1;
   return 3 + 4;
   a[i] = b[j];
   return;
   while (TRUE) {a = 3; b = 4; }
   if (a == b) f (x);
   if (a != b) y = 3; else f (y, z);
}

```

```
/*
This is an unterminated comment.
It would cause cpp to error out.
When cpp returns a non-zero exit code,
so should your compiler.
$Id: 92-uncomment.oc,v 1.1 2019-05-09 14:45:22-07 - - $

int main (int argc, char** argv) {

   Your compiler never sees any of this code.

}

It should notice the incorrect return status from cpp.
```

```
// $Id: 93-semantics.oc,v 1.1 2019-05-09 14:45:22-07 - - $
// This code should scan and parse correctly,
// but fail to type check.
int[] a = null;
int[] b = null;

void[] f() {}; // can't have void[]

int main() {
   int c = a + b; // can't add arrays
   void n = null; // can't have void vars
   int x = a < b; // can't compare pointers <
   int y = a==b; // this is ok
   return "foobar";
}

```

```
// $Id: 94-syntax.oc,v 1.1 2019-05-09 14:45:22-07 - - $

int f() {
int a = ;
return foo;
public static int main (String[] args) {
   System.exit (255);
}

```

```
// $Id: 95-cobol.oc,v 1.1 2019-05-09 14:45:22-07 - - $

000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.     HELLOWORLD.
000300
000400*
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400     DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500     DISPLAY "Hello world!" LINE 15 POSITION 10.
100600     STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800     EXIT.
```

```
// Unterminated strings.
// $Id: 96-unterminated.oc,v 1.1 2019-05-09 14:45:22-07 - - $

int main() {
   string t = "\*/";
   string s = "abc;
   char c = 'a;
   s = "abcd\";
   s = "abc|\
   ;
   int 23foobar;
}

```

```
# $Id: Makefile,v 1.21 2019-10-11 14:21:09-07 - - $

UTILDIR = /afs/cats.ucsc.edu/courses/cse110a-wm/bin

NOWARN  = -Wno-write-strings -Wno-main
OCGPP   = g++ ${NOWARN} -x c++ -include octypes.h

OCSRC   = ${sort ${wildcard [0-8]*.oc}}
LIBSRC  = oclib.c
LIBOBJ  = ${LIBSRC:.c=.o}
HEADERS = oclib.h octypes.h
OCOBJ   = ${OCSRC:.oc=.o}
EXECS   = ${OCSRC:.oc=}
ALLSRC  = ${OCSRC} 9*.oc Makefile ${HEADERS} ${LIBSRC}
LISTING = Listing.oc-programs

all : ${EXECS}

% : %.o ${LIBOBJ}
        g++ $< ${LIBOBJ} -o $@

%.o : %.oc ${HEADERS}
        - cid + $<
        - checksource $<
        ${OCGPP} -c $<

${LIBOBJ} : ${LIBSRC}
        - cid + $<
        - checksource $<
        gcc -c $<

spotless : clean
        - rm ${LISTING}.{ps,pdf} ${EXECS}

clean :
        -rm ${OCOBJ} ${LIBOBJ} oclib.nm Listing.asm.{ps,pdf}

ci :
        ${UTILDIR}/cid + ${ALLSRC}

lis :
        ${UTILDIR}/checksource ${ALLSRC}
        ${UTILDIR}/mkpspdf ${LISTING}.ps ${ALLSRC}

asm : ${LIBOBJ}
        nm -a ${LIBOBJ} >oclib.nm
        ${UTILDIR}/mkpspdf Listing.asm.ps \
              oclib.h oclib.c oclib.nm oclib.s

again :
        make --no-print-directory clean ci all lis
```

```
// $Id: oclib.h,v 1.13 2019-09-19 17:08:34-07 - - $

// Bilingual file useable as a header file for both oc and g++.

#ifndef __OCLIB_H__
#define __OCLIB_H__

#ifdef __cplusplus
extern "C" {
using string = char*;
#endif

#define SUCCESS 0
#define FAILURE 1
#define BOOL int
#define TRUE 1
#define FALSE 0
#define EOF (-1)

#define assert(expr) {if (not (expr)) fail (#expr, __FILE__, __LINE__);}

void fail (string expr, string file, int line);

void putchr (int chr);
void putint (int num);
void putstr (string str);

int getchr();
string getstr();
string getln();

#ifdef __cplusplus
}
#endif

#endif

```

```
// $Id: octypes.h,v 1.4 2019-09-16 14:36:17-07 - - $

// Type definitiions to compile oc programs with g++.

#ifndef __OCDEFS_H__
#define __OCDEFS_H__

#include <type_traits>

using string = char*;

template <typename type>
using ptr = std::enable_if_t<std::is_class<type>::value,type*>;

template <typename type>
struct array {
   using array_value_type = type;
   type* data {};
   array() = default;
   array (type* that) { data = that; }
   array& operator= (type* that) { data = that; return *this; }
   type& operator[] (int i) { return data[i]; }
};

template <typename type>
std::enable_if_t<std::is_class<type>::value,ptr<type>>
alloc() {
   return new type();
}

template <typename type>
array<typename type::array_value_type>
alloc (int size) {
   auto result = new typename type::array_value_type [size] {};
   using result_t = array<typename type::array_value_type>*;
   return *reinterpret_cast<result_t> (&result);
}

template <typename type>
std::enable_if_t<std::is_same<type,string>::value,string>
alloc (int size) {
   return new char[size] {};
}

#endif

```

```c
// $Id: oclib.c,v 1.7 2019-09-19 17:08:34-07 - - $

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define not !
#define nullptr 0
#define string char*

#include "oclib.h"

void fail (string expr, string file, int line) {
   fprintf (stderr, "%s:%d: assert (%s) failed\n", file, line, expr);
   abort();
}

void* xcalloc (int nelem, int size) {
   void* result = calloc (nelem, size);
   assert (result != nullptr);
   return result;
}

void putchr (int chr) { printf ("%c", chr); }
void putint (int num) { printf ("%d", num); }
void putstr (string str) { printf ("%s", str); }

int getchr() { return getchar(); }

static char get_buffer[0x1000];

string getstr (void) {
   static char format[16];
   sprintf (format, "%%%zds", sizeof get_buffer - 1);
   int count = scanf (format, get_buffer);
   return count != 1 ? nullptr : strdup (get_buffer);
}

string getln (void) {
   string result = fgets (get_buffer, sizeof get_buffer, stdin);
   return result == nullptr ? nullptr : strdup (result);
}

```