Hai.java

1: class Hai
2: {
3:     public static void main(String [] args)
4:     {
5:         System.out.println("O hai, world!");
6:     }
7: }

argv1.c

#include <stdio.h>

int main(int argc, char *argv[])

{ int i;
  int c = 0;

  for (i = 0; i < argc; i++)
  {
    printf("%s
", argv[i]);
    printf("%s
", argv[i]);
    printf("\n");
  }

  return 0;
}
argv2.c

1: /*********************************************************************************
2:  * argv2.c
3:  *
4:  * Computer Science 50
5:  * David J. Malan
6:  *
7:  * Prints command-line arguments, one character per line.
8:  *
9:  * Demonstrates argv as a two-dimensional array.
10:  *********************************************************************************/
11: 12: #include <stdio.h>
13: #include ... 
14: 15: 16: int
17: main(int argc, char *argv[])
18: { 19:     int i, j, n;
20:     // print arguments
21:     printf("\n");
22:     for (i = 0; i < argc; i++)
23:     {
24:         for (j = 0, n = strlen(argv[i]); j < n; j++)
25:             printf("%c
", argv[i][j]);
26:     }
27:     printf("\n");
28: } 29: }

array.c

1: /*********************************************************************************
2:  * array.c
3:  *
4:  * Computer Science 50
5:  * David J. Malan
6:  *
7:  * Computes a student's average across 3 quizzes (without dropping lowest).
8:  *
9:  * Demonstrates use of an array, a constant, a multiline string, and rounding.
10:  *********************************************************************************/
11: 12: #include <cs50.h>
13: #include <stdio.h>
14: 15: 16: int
17: main(int argc, char *argv[])
18: { 19:     int
20:     float grades[QUIZZES];
21:     double average;
22:     int i, sum;
23:     printf("What were your quiz scores? \n\n");
24:     for (i = 0; i < QUIZZES; i++)
25:     {
26:         printf("Quiz \#%d of \#%d: ", i+1, QUIZZES);
27:         grades[i] = GetFloat();
28:     }
29:     sum = 0;
30:     for (i = 0; i < QUIZZES; i++)
31:     {
32:         sum += grades[i];
33:     }
34:     average = (int) (sum / (double) QUIZZES + 0.5);
35:     printf("\nWithout dropping your lowest score, 
36:     "your average is: \d\n\n", average); 37: }
#array1.c

```c
#include <cs50.h>
#include <stdio.h>

#define QUIZZES 2

int main(int argc, char *argv[])
{
    float grades[QUIZZES], sum; // declare float array
    // ask user for grades
    printf("
What were your quiz scores?\n\n*");
    for (i = 0; i < QUIZZES; i++)
    {
        printf("Quiz \#%d of \#%d: ", i+1, QUIZZES);
        grades[i] = GetFloat();
    }
    // compute average
    sum = 0;
    for (i = 0; i < QUIZZES; i++)
    {
        sum += grades[i];
    }
    average = (int) (sum / QUIZZES + 0.5);
    // report average
    printf("\nYour average is: \d\n\n", average);
    return 0;
}
```

#array2.c

```c
#include <cs50.h>
#include <math.h>

#define QUIZZES 2

int main(int argc, char *argv[])
{
    float grades[QUIZZES], sum; // declare float array
    int average, i;
    // ask user for grades
    printf("What were your quiz scores?\n\n*");
    for (i = 0; i < QUIZZES; i++)
    {
        printf("Quiz \#%d of \#%d: ", i+1, QUIZZES);
        grades[i] = GetFloat();
    }
    // compute average
    sum = 0;
    for (i = 0; i < QUIZZES; i++)
    {
        sum += grades[i];
    }
    average = (int) round(sum / QUIZZES);
    // report average
    printf("\nThe average is: \d\n\n", average);
    return 0;
}
```
ascii1.c

1: /**************************************************************************
2:  * ascii1.c3:  *4:  * Computer Science 50
5:  * David J. Malan6:  *7:  * Displays the mapping between alphabetical ASCII characters and
8:  * their decimal equivalents using one column.
9:  *10:  * Demonstrates casting from int to char.
11: /**************************************************************************/
12:
13: #include <stdio.h>
14:
15:
16: int
17: main(int argc, char *argv[])
18: {19:  int i;
20:     // display mapping for uppercase letters
21:     for (i = 65; i < 65 + 26; i++)
22:         printf("%c %d \n", (char) i, i);
23:
24:     // separate uppercase from lowercase
25:     printf("\n");
26:
27:     // display mapping for lowercase letters
28:     for (i = 97; i < 97 + 26; i++)
29:         printf("%c %d \n", (char) i, i);
30:
31: }
32:

ascii2.c

1: /**************************************************************************
2:  * ascii2.c3:  *4:  * Computer Science 50
5:  * David J. Malan6:  *7:  * Displays the mapping between alphabetical ASCII characters and
8:  * their decimal equivalents using two columns.
9:  *10:  * Demonstrates specification of width in format string.
11: /**************************************************************************/
12:
13: #include <stdio.h>
14:
15:
16: int
17: main(int argc, char *argv[])
18: {19:  int i;
20:     // display mapping for uppercase letters
21:     for (i = 65; i < 65 + 26; i++)
22:         printf("%c %d %3d %c\n", (char) i, i, i + 32, (char) (i + 32));
23:
24: }
25:
ascii3.c
************
3: * ascii3.c
4: * Computer Science 50
5: * David J. Malan
6: *
7: * Displays the mapping between alphabetical ASCII characters and
8: * their decimal equivalents.
9: *
10: * Demonstrates iteration with a char.
11: ***********************************************/
12:
13: #include <stdio.h>
14:
15:
16: int
17: main(int argc, char *argv[])
18: {
19:      char c;
20:     // display mapping for uppercase letters
21:     for (c = 'A'; c <= 'Z'; c = (char) ((int) c + 1))
22:         printf("%c: %d\n", c, (int) c);
23: }
24: }

battleship.c
************
3: * battleship.c
4: * Computer Science 50
5: * David J. Malan
6: *
7: * Prints a Battleship board.
8: *
9: * Demonstrates nested loop.
10: ***********************************************/
11:
12: #include <stdio.h>
13:
14:
15: int
16: main(int argc, char *argv[])
17: {
18:      int i, j;
19:     // print top row of numbers
20:     printf("\n ");
21:     for (i = 1; i <= 10; i++)
22:         printf("%d ", i);
23:     printf("\n ");
24:     printf("\n ");
25:     // print rows of holes, with letters in leftmost column
26:     for (i = 0; i < 10; i++)
27:     {
28:         printf("%c ", 'A' + i);
29:         for (j = 1; j <= 10; j++)
30:             printf("\n ");
31:         printf("\n ");
32:     }
33:     printf("\n ");
34: }
beer1.c

1: //****************************************************************************
2:  * beer1.c
3:  *
4:  * Computer Science 50
5:  *
6:  *
7:  * Sings "99 Bottles of Beer on the Wall."
8:  *
9:  * Demonstrates a for loop (and an opportunity for hierarchical
10:  * decomposition).
11:  ****************************************************************************/
12: #include <stdio.h>
13: #include <cs50.h>
14: 15: 16: 17: int
18: main(int argc, char *argv[])
19: {
20:     int i, n;
21: 22:     // ask user for number
23:     printf("How many bottles will there be? ");
24:     n = GetInt();
25: 26:     // exit upon invalid input
27:     if (n < 1)
28:     {
29:         printf("Sorry, that makes no sense.\n");
30:         return 1;
31:     }
32: 33: // sing the annoying song
34:     printf("\n");
35:     for (i = n; i > 0; i--)
36:     {
37:         printf("%d bottle(s) of beer on the wall,\n", i);
38:         printf("%d bottle(s) of beer,\n", i);
39:         printf("Take one down, pass it around,\n");
40:         printf("%d bottle(s) of beer on the wall.\n", i - 1);
41:     }
42: 43: // exit when song is over
44:     printf("Wow, that's annoying.\n");
45:     return 0;
46: }

beer2.c

1: //****************************************************************************
2:  * beer2.c
3:  *
4:  * Computer Science 50
5:  *
6:  *
7:  * Sings "99 Bottles of Beer on the Wall."
8:  *
9:  * Demonstrates a while loop (and an opportunity for hierarchical
10:  * decomposition).
11:  ****************************************************************************/
12: #include <stdio.h>
13: #include <cs50.h>
14: 15: 16: 17: int
18: main(int argc, char *argv[])
19: {
20:     int n;
21: 22:     // ask user for number
23:     printf("How many bottles will there be? ");
24:     n = GetInt();
25: 26:     // exit upon invalid input
27:     if (n < 1)
28:     {
29:         printf("Sorry, that makes no sense.\n");
30:         return 1;
31:     }
32: 33: // sing the annoying song
34:     printf("\n");
35:     while (n > 0)
36:     {
37:         printf("%d bottle(s) of beer on the wall,\n", n);
38:         printf("%d bottle(s) of beer,\n", n);
39:         printf("Take one down, pass it around,\n");
40:         printf("%d bottle(s) of beer on the wall.\n", n - 1);
41:         n--;
42:     }
43: 44: // exit when song is over
45:     printf("Wow, that's annoying.\n");
46:     return 0;
47: }
beer3.c

1: /**************************************************************************
2:  * beer3.c
3:  * Computer Science 50
4:  * David J. Malan
5:  * Sings "99 Bottles of Beer on the Wall."
6:  * Demonstrates a condition within a for loop.
7:  ***************************************************************************/
8: #include <cs50.h>
9: #include <stdio.h>
10: int main(int argc, char *argv[])
11: {
12:     int i, n;
13:     string s1, s2;
14:     // ask user for number
15:     printf("How many bottles will there be? ");
16:     n = GetInt();
17:     // exit upon invalid input
18:     if (n < 1)
19:     { 
20:         printf("Sorry, that makes no sense.\n");
21:         return 1;
22:     }
23:     // sing the annoying song
24:     printf("%d %s of beer on the wall,\n", n, s1);
25:     for (i = n; i > 0; i--)
26:     { 
27:         // use proper grammar
28:         s1 = (i == 1) ? "bottle" : "bottles";
29:         s2 = (i == 2) ? "bottle" : "bottles";
30:         // sing verses
31:         printf("%d %s of beer on the wall,\n", i, s1);
32:         printf("%d %s of beer on the wall,\n", i - 1, s2);
33:         // exit when song is over
34:         printf("Wow, that's annoying.\n");
35:         return 0;
36:     }
37: }
38: #include <cs50.h>
39: #include <stdio.h>
40: int main(int argc, char *argv[])
41: {
42:     int n;
43:     // ask user for number
44:     printf("How many bottles will there be? ");
45:     n = GetInt();
46:     // exit upon invalid input
47:     if (n < 1)
48:     { 
49:         printf("Sorry, that makes no sense.\n");
50:         return 1;
51:     }
52:     // sing the annoying song
53:     printf("%d %s of beer on the wall,\n", n, s1);
54:     for (i = n; i > 0; i--)
55:     { 
56:         // use proper grammar
57:         s1 = (i == 1) ? "bottle" : "bottles";
58:         s2 = (i == 2) ? "bottle" : "bottles";
59:         // sing verses
60:         printf("%d %s of beer on the wall,\n", n, s1);
61:         printf("%d %s of beer on the wall,\n", i, s1);
62:         printf("Take one down, pass it around,\n");
63:         printf("%d %s of beer on the wall,\n", i - 1, s2);
64:         // exit when song is over
65:         printf("Wow, that's annoying.\n");
66:         return 0;
67:     }
68: #include <cs50.h>
69: #include <stdio.h>
70: int main(int argc, char *argv[])
71: {
72:     int n;
73:     // ask user for number
74:     printf("How many bottles will there be? ");
75:     n = GetInt();
76:     // exit upon invalid input
77:     if (n < 1)
78:     { 
79:         printf("Sorry, that makes no sense.\n");
80:         return 1;
81:     }
82:     // sing the annoying song
83:     printf("%d %s of beer on the wall,\n", n, s1);
84:     for (i = n; i > 0; i--)
85:     { 
86:         // use proper grammar
87:         s1 = (i == 1) ? "bottle" : "bottles";
88:         s2 = (i == 2) ? "bottle" : "bottles";
89:         // sing verses
90:         printf("%d %s of beer on the wall,\n", n, s1);
91:         printf("%d %s of beer on the wall,\n", i, s1);
92:         printf("Take one down, pass it around,\n");
93:         printf("%d %s of beer on the wall,\n", i - 1, s2);
94:         // exit when song is over
95:         printf("Wow, that's annoying.\n");
96:         return 0;
97:     }
98: }
beer4.c

65:     printf("%d %s of beer,\n", b, s1);
66:     printf("Take one down, pass it around,\n");
67:     printf("%d %s of beer on the wall.\n\n", b - 1, s2);
68: }

buggy1.c

1: /******************************************************************************/
2: * buggy1.c
3: *
4: * Computer Science 50
5: * David J. Malan
6: *
7: * Should print 10 asterisks but doesn't!
8: * Can you find the bug?
9: ******************************************************************************/
10: #include <stdio.h>
11: int
12: main(int argc, char *argv[])
13: { int i;
14:     for (i = 0; i <= 10; i++)
15:         printf("*\n");
16: }

buggy2.c

1: /**************************************************************************
2: * buggy2.c
3: *
4: * Computer Science 50
5: * David J. Malan
6: *
7: * Should print 10 asterisks, one per line, but doesn't!
8: * Can you find the bug?
9: **************************************************************************/
10: #include <stdio.h>
11: int
12: main(int argc, char *argv[])
13: { int i;
14:     for (i = 0; i <= 10; i++)
15:         printf("*");
16:         printf("\n");
17: }

buggy3.c

1: /**************************************************************************
2: * buggy3.c
3: *
4: * Computer Science 50
5: * David J. Malan
6: *
7: * Should swap two variables' values, but doesn't!
8: * Can you find the bug?
9: **************************************************************************/
10: #include <stdio.h>
11: int
12: main(int argc, char *argv[])
13: { int
14:     int a, int b;
15:     printf("x is \n", x);
16:     printf("y is \n", y);
17:     swap(x, y);
18:     printf("swapped\n!n");
19:     printf("x is \n", x);
20:     printf("y is \n", y);
21: }
22: void
23: swap(int a, int b)
24: { int tmp;
25:     tmp = a;
26:     a = b;
27:     b = tmp;
28: }
#include <stdio.h>

void increment() {
    x++;  // Tries to increment x.
}

int main(int argc, char *argv[]) {
    int x = 10;
    x++;  // Increments x.
    printf("\n\nx is now %d\n", x);
    return 0;
}

/*
 * void increment()
 * x++;
 */
buggy6.c

1: /**************************************************************************
2: * buggy6.c
3: *
4: * Computer Science 50
5: * David J. Malan
6: *
7: * Asks student for their grades but prints too many!
8: * Can you find the bug?
9: *
10: * Demonstrates accidental use of a "magic number."
11: **************************************************************************/
12: #include <cs50.h>
13: #include <stdio.h>
14: #define QUIZZES 3
15:
16:
17: // number of quizzes per term
18: int main(int argc, char *argv[])
19: {
20:     int i; // ask user for scores
21:     float grades[QUIZZES];
22:     for (i = 0; i < QUIZZES; i++)
23:         grades[i] = GetFloat();
24:     // print scores
25:     for (i = 0; i < 3; i++)
26:         printf("%.2f\n", grades[i]);
27: }
28: #endif
29: #endif
30: #endif
31: #endif
32: #endif
33: #endif
34: #endif
35: #endif
36: #endif
37: #endif
38: #endif
39: #endif
40: #endif
41: #endif
42: #endif
43: #endif
44: #endif
45: #endif
46: #endif
47: #endif
48: #endif
49: #endif
50: #endif
51: #endif
52: #endif
53: #endif
54: #endif
55: #endif
56: #endif
57: #endif
58: #endif
59: #endif
60: #endif
61: #endif
62: 
63: capitalize.c

1: /**************************************************************************
2: * capitalize.c
3: *
4: * Computer Science 50
5: * David J. Malan
6: *
7: * Capitalizes a given string.
8: *
9: * Demonstrates casting and iteration over strings as arrays of chars.
10: **************************************************************************/
11: #include <cs50.h>
12: #include <stdio.h>
13: #include <string.h>
14: #define MAXCHARS 20
15:
16:
17: int main(int argc, char *argv[])
18: { int i, n;
19:     string s;
20:     // get line of text
21:     s = GetString();
22:     // capitalize text
23:     for (i = 0, n = strlen(s); i < n; i++)
24:     {
25:         if (s[i] >= 'a' && s[i] <= 'z')
26:             printf("%c", s[i] - ('a' - 'A'));
27:         else
28:             printf("%c", s[i]);
29:     }
30:     printf("\n");
31: }
Definitions for CS 50's library.

Based on Eric Roberts' genlib.c and simpio.c.

The latest version of this file can be found at http://cs50.net/pub/releases/cs50/cs50.c.

To compile as a static library on your own system:
% gcc -c -gdbs -std=99 cs50.c -o cs50.o
% rm -f cs50.o
% cp cs50.h /usr/local/include
% cp libcs50.a /usr/local/lib

******************************************************************************

---

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "cs50.h"

#define CAPACITY 128

char *GetChar();

float GetFloat();

---

float GetFloat()
{
    return c1;
}

else
{
    free(line);
    printf("Retry: ");
}

/*
  * Reads a line of text from the user, returning the equivalent.
  * If line can't be read, returns DBL_MAX.
  */

float GetFloat()
{
    // try to get a double from user
    while (true)
    {
        // get line of text, returning DBL_MAX on failure
        string line = GetString();
        if (line == NULL)
        {
            return CHAR_MAX;
        }
        // return a double if only a double (possibly with
        // leading and/or trailing whitespace) was provided
        double d; char c;
        if (sscanf(line, "%lf %c", &d, &c) == 1)
        {
            free(line);
            return d;
        }
        else
        {
            free(line);
            printf("Retry: ");
        }
    }
}
129:     // try to get a float from user
130:     while (true) {
131:         // get line of text, returning FLT_MAX on failure
132:         string line = GetString();
133:         if (line == NULL) return FLT_MAX;
134:     }
135:     // return a float if only a float (possibly with leading and/or trailing whitespace) was provided
136:     char c; float f;
137:     if (sscanf(line, " \f %c\", &f, &c) == 1) {
138:         free(line);
139:         return f;
140:     }
141:     else {
142:         free(line);
143:         printf("Retry: ");
144:     }
145: }
146: /*
147: * int
148: * GetInt()
149: */
150: /*
151: * Reads a line of text from standard input and returns it as an int in the range of [-2^31 + 1, 2^31 - 1], if possible; if text
152: * does not represent such an int, user is prompted to retry. Leading
153: * and trailing whitespace is ignored. For simplicity, overflow is not
154: * detected. If line can't be read, returns INT_MAX.
155: */
156: int
157: GetInt() {
158:     // try to get an int from user
159:     while (true) {
160:         // get line of text, returning INT_MAX on failure
161:         string line = GetString();
162:         if (line == NULL) return INT_MAX;
163:     }
164:     // return an int if only an int (possibly with leading and/or trailing whitespace) was provided
165:     int n; char c;
166:     if (sscanf(line, " \d %c", &n, &c) == 1) {
167:         free(line);
168:         return n;
169:     } else {
170:         free(line);
171:         printf("Retry: ");
172:     }
173: /*
174: * string
175: * GetString()
176: */
177: string
178: GetString() {
179:     // reads a line of text from standard input and returns it as a string,
180:     // with trailing newline character. Ergo, if user inputs only "\n",
181:     // returns "\n" not NULL. Leading and trailing whitespace is not ignored.
182:     // Returns NULL upon error or no input whatsoever (i.e., just EOF).
183:     */
184: string
185: GetString() {
186:     // growable buffer for chars
187:     string buffer = NULL;
188:     // capacity of buffer
189:     unsigned int capacity = 0;
190:     // number of chars actually in buffer
191:     unsigned int n = 0;
192:     // character read or EOF
193:     int c;
// iteratively get chars from standard input
while ( (c = fgetc(stdin)) != '
' && c != EOF) {
    // grow buffer if necessary
    if (n + 1 > capacity)
        { // determine new capacity: start at CAPACITY then double
            if (capacity == 0)
                capacity = CAPACITY;
            else if (capacity == (UINT_MAX / 2))
                capacity *= 2;
            else
                { free(buffer);
                  return NULL;
                }
            // extend buffer's capacity
            string temp = realloc(buffer, capacity * sizeof(char));
            if (temp == NULL)
                { free(buffer);
                  return NULL;
                }
            buffer = temp;
        }
    // append current character to buffer
    buffer[n++] = c;
}
// return NULL if user provided no input
if (n == 0 && c == EOF)
    return NULL;
// minimize buffer
string minimal = malloc((n + 1) * sizeof(char));
strncpy(minimal, buffer, n);
free(buffer);
// terminate string
minimal[n] = '\0';
return minimal;
}

// declarations for CS 50's library.
// Based on Eric Roberts' genlib.h and simpio.h.
// The latest version of this file can be found at
// http://cs50.net/pub/releases/cs50/cs50.h.
// To compile as a static library on your own system:
# gcc -c -gdwarf -std=c99 cs50.c -o cs50.o
# ar rcs libcs50.a cs50.o
# rm -f cs50.o
# cp cs50.h /usr/local/include
# cp libcs50.a /usr/local/lib

/*
 * double
 */

typedef char *string;

/*
 * char
 */

// Reads a line of text from standard input and returns the equivalent
// char; if text does not represent a char, user is prompted to retry.
// Leading and trailing whitespace is ignored. If line can't be read, returns CHAR_MAX.

char GetChar();

char GetChar();
65:  * GetDouble()
66:  *
67:  * Reads a line of text from standard input and returns the equivalent
68:  * double as precisely as possible; if text does not represent a
double, user is prompted to retry. Leading and trailing whitespace
70:  * is ignored. For simplicity, overflow and underflow are not detected.
71:  * If line can't be read, returns DBL_MAX.
72:  */
73: 
74:  double GetDouble();
75: 
76: 77:  /*
78:  * float
79:  * GetFloat()
80:  *
81:  * Reads a line of text from standard input and returns the equivalent
82:  * float as precisely as possible; if text does not represent a float,
83:  * user is prompted to retry. Leading and trailing whitespace is ignored.
84:  * For simplicity, overflow and underflow are not detected. If line can't
85:  * be read, returns FLT_MAX.
86:  */
87: 
88:  float GetFloat();
89: 
90: 91:  /*
92:  * int
93:  *GetInt()
94:  *
95:  * Reads a line of text from standard input and returns it as an
96:  * int in the range [-2^31 + 1, 2^31 - 2], if possible; if text
97:  * does not represent such an int, user is prompted to retry. Leading
98:  * and trailing whitespace is ignored. For simplicity, overflow is not
99:  * detected. If line can't be read, returns INT_MAX.
100: */
101: 
102: int GetInt();
103: 
104: 105:  /*
106:  * long long
107:  * GetLongLong()
108:  *
109:  * Reads a line of text from standard input and returns an equivalent
110:  * long long in the range [-2^63 + 1, 2^63 - 2], if possible; if text
111:  * does not represent such a long long, user is prompted to retry.
112:  * Leading and trailing whitespace is ignored. For simplicity, overflow
113:  * is not detected. If line can't be read, returns LONGLONG_MAX.
114: */
115: 
116: long long GetLongLong();
117: 
118: 119: /*
120:  * string
121:  * GetString()
122:  *
123:  * Reads a line of text from standard input and returns it as a string,
124:  * sans trailing newline character. (Ryo, if user inputs only "\n",
125:  * returns ** not NULL.) Leading and trailing whitespace is not ignored.
126:  * Returns NULL upon error or no input whatsoever (i.e., just EOF).
127:  */
128:
global.c

```c
#include <stdio.h>

// ... /*
* void
* increment()
* */

void increment() {
    x++;
}
```
hai.lisp
lectures/2/src/
1: (print "O hai, world!")

hai.php
lectures/2/src/
1: <?
2:     echo "O hai, world!\n";
3: ?>
hai.pl
lectures/2/src/
1: MAIN:
2: { 
3:  print "O hai, world!\n";
4: }

return1.c
lectures/2/src/
1: /**************************************************************************
2: * return1.c
3: *
4: * Computer Science 50
5: * David J. Malan
6: *
7: * Increments a variable.
8: *
9: * Demonstrates use of parameter and return value.
10: ***************************************************************************/
11: 12: #include <stdio.h>
13: 14: 15: // function prototype
16: int increment(int);
17: 18: 19: int
20: main(int argc, char *argv[])
21: { 
22:  int x = 1;
23:  printf("x is now %d\n", x);
24:  printf("Incrementing...\n");
25:  x = increment(x);
26:  printf("Incremented\n");
27:  printf("x is now %d\n", x);
28: }
29: 30: 31: /*
32: *  
33: *  int
34: *  increment(int a)
35: *  */
36: 37: int
38: increment(int a)
39: { 
40:  return a + 1;
41:  
42: }
return2.c

```c
#include <stdio.h>

int cube(int a) {
    return a * a * a;
}
```

string1.c

```c
#include <cs50.h>
#include <stdio.h>

int cube(int a) {
    return a * a * a;
}
```
#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
    int i, n;
    string s;
    // get line of text
    s = GetString();
    // print string, one character per line
    for (i = 0, n = strlen(s); i < n; i++)
    {
        printf("%c
", s[i]);
    }
    return 0;
}