/# argv1.c
* Computer Science 50
* David J. Malan
* Prints command-line arguments, one per line.
* Demonstrates use of argv.
***************************************************************************/

#include <stdio.h>

int main(int argc, char *argv[])
{
    // print arguments
    printf("\n");
    for (int i = 0; i < argc; i++)
        printf("%s\n", argv[i]);
    printf("\n");
}
# argv2.c

* Computer Science 50
* David J. Malan
* Prints command-line arguments, one character per line.
* Demonstrates argv as a two-dimensional array.

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

int main(int argc, char *argv[])
{
    // print arguments
    printf("\n");
    for (int i = 0; i < argc; i++)
    {
        for (int j = 0, n = strlen(argv[i]); j < n; j++)
            printf("%c\n", argv[i][j]);
        printf("\n");
    }
}
#include <cs50.h>
#include <math.h>
#include <stdio.h>

#define QUIZZES 2

int main(void)
{
    // ask user for grades
    float grades[QUIZZES];
    printf("What were your quiz scores?\n\n");
    for (int i = 0; i < QUIZZES; i++)
    {
        printf("Quiz %d of %d: ", i+1, QUIZZES);
        grades[i] = GetFloat();
    }

    // compute average
    float sum = 0;
    for (int i = 0; i < QUIZZES; i++)
    {
        sum += grades[i];
    }
    int average = (int) round(sum / QUIZZES);

    // report average
    printf("Your average is: %d\n\n", average);
}
#include <cs50.h>
#include <stdio.h>

// number of quizzes per term
#define QUIZZES 2

int main(void)
{
    // ask user for grades
    float grades[QUIZZES];
    printf("What were your quiz scores?\n\n");
    for (int i = 0; i < QUIZZES; i++)
    {
        printf("Quiz #\d of #d: ", i+1, QUIZZES);
        grades[i] = GetFloat();
    }
    // compute average
    float sum = 0;
    for (int i = 0; i < QUIZZES; i++)
    {
        sum += grades[i];
    }
    int average = (int) (sum / QUIZZES + 0.5);
    // report average
    printf("Your average is: %d\n\n", average);
}
/*beer1.c
 * Computer Science 50
 * David J. Malan
 * Sings "99 Bottles of Beer on the Wall."
 * Demonstrates a for loop (and an opportunity for hierarchical
decomposition).
****************************************************************************/

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

int main(void)
{
    // ask user for number
    printf("How many bottles will there be? ");
    int n = GetInt();

    // exit upon invalid input
    if (n < 1)
    {
        printf("Sorry, that makes no sense.\n");
        return 1;
    }

    // sing the annoying song
    printf("\n");
    for (int i = n; i > 0; i--)
    {
        printf("%d bottle(s) of beer on the wall,\n", i);
        printf("%d bottle(s) of beer,\n", i);
        printf("Take one down, pass it around,\n");
        printf("%d bottle(s) of beer on the wall.\n\n", i - 1);
    }

    // exit when song is over
    printf("Wow, that's annoying.\n");
    return 0;
}
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // ask user for number
    printf("How many bottles will there be? ");
    int n = GetInt();
    // exit upon invalid input
    if (n < 1)
    {
        printf("Sorry, that makes no sense.\n");
        return 1;
    }
    // sing the annoying song
    printf("\n");
    while (n > 0)
    {
        printf("%d bottle(s) of beer on the wall,\n", n);
        printf("%d bottle(s) of beer,\n", n);
        printf("Take one down, pass it around,\n");
        printf("%d bottle(s) of beer on the wall.\n\n", n - 1);
        n--;
    }
    // exit when song is over
    printf("Wow, that's annoying.\n");
    return 0;
}
/* beer3.c */
* Computer Science 50
* David J. Malan
* Sings "99 Bottles of Beer on the Wall."
* Demonstrates a condition within a for loop.
***************************************************************************/

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

int main(void)
{
    // ask user for number
    printf("How many bottles will there be? ");
    int n = GetInt();

    // exit upon invalid input
    if (n < 1)
    {
        printf("Sorry, that makes no sense.\n");
        return 1;
    }

    // sing the annoying song
    printf("\n");
    for (int i = n; i > 0; i--)
    {
        // use proper grammar
        string s1 = (i == 1) ? "bottle" : "bottles";
        string s2 = (i == 2) ? "bottle" : "bottles";

        // sing verses
        printf("%d %s of beer on the wall,\n", i, s1);
        printf("%d %s of beer,\n", i, s1);
        printf("Take one down, pass it around,\n");
        printf("%d %s of beer on the wall.\n\n", i - 1, s2);
    }

    // exit when song is over
    printf("Wow, that's annoying.\n");
    return 0;
}
* beer4.c
* Computer Science 50
* David J. Malan
* Sings "99 Bottles of Beer on the Wall."
* Demonstrates hierarchical decomposition and parameter passing.

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

// function prototype
void chorus(int b);

int main(void) {
    // ask user for number
    printf("How many bottles will there be? ");
    int n = GetInt();

    // exit upon invalid input
    if (n < 1) {
        printf("Sorry, that makes no sense.\n");
        return 1;
    }

    // sing the annoying song
    printf("\n");
    while (n) {
        chorus(n--);
    }

    // exit when song is over
    printf("Wow, that's annoying.\n");
    return 0;
}
```

* Sings about specified number of bottles.
*
void chorus(int b)
{
    // use proper grammar
    string s1 = (b == 1) ? "bottle" : "bottles";
    string s2 = (b == 2) ? "bottle" : "bottles";

    // sing verses
    printf("%d %s of beer on the wall,
", b, s1);
    printf("%d %s of beer,
", b, s1);
    printf("Take one down, pass it around,
")
    printf("%d %s of beer on the wall.
", b - 1, s2);
}

# include <stdio.h>

// function prototype
void swap(int a, int b);

int main(void)
{
    int x = 1;
    int y = 2;

    printf("x is %d\n", x);
    printf("y is %d\n", y);
    printf("Swapping...\n");
    swap(x, y);
    printf("Swapped!\n");
    printf("x is %d\n", x);
    printf("y is %d\n", y);
}

/*
 * Swap arguments' values.
 */

void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
#include <stdio.h>

// function prototype
void increment(void);

int main(void)
{
    int x = 1;
    printf("x is now %d\n", x);
    printf("Incrementing...\n");
    increment();
    printf("Incremented!\n");
    printf("x is now %d\n", x);
}

/*
 * Tries to increment x.
 */

void increment(void)
{
    x++;
}
#include <stdio.h>

int x;

void increment(void);

int main(void)
{
    printf("x is now %d\n", x);
    printf("Initializing...\n");
    x = 1;
    printf("Initialized!\n");
    printf("x is now %d\n", x);
    printf("Incrementing...\n");
    increment();
    printf("Incremented!\n");
    printf("x is now %d\n", x);
}

/*
* Increments x.
*/

void increment(void)
{
    int x = 10;
    x++;
}
#include <cs50.h>
#include <stdio.h>

// number of quizzes per term
#define QUIZZES 2

int main(void)
{
    float grades[QUIZZES];
    // ask user for scores
    printf("What were your quiz scores?\n\n");
    for (int i = 0; i < QUIZZES; i++)
    {
        printf("Quiz #%d of %d: ", i+1, QUIZZES);
        grades[i] = GetFloat();
    }
    // print scores
    for (int i = 0; i < 3; i++)
        printf("%.2f\n", grades[i]);
#ifndef _CS50_H
#define _CS50_H

#include <float.h>
#include <limits.h>
#include <stdbool.h>
#include <stdlib.h>

/*
 * Our own data type for string variables.
 */
typedef char *string;

/*
 * 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(void);

/*
 * Reads a line of text from standard input and returns the equivalent
 * double as precisely as possible; if text does not represent a
double, user is prompted to retry. Leading and trailing whitespace
* is ignored. For simplicity, overflow and underflow are not detected.
* If line can't be read, returns DBL_MAX.
*/
double GetDouble(void);

float GetFloat(void);

int GetInt(void);

long long GetLongLong(void);

/*
 * Reads a line of text from standard input and returns the equivalent
 * float as precisely as possible; if text does not represent a float,
 * user is prompted to retry. Leading and trailing whitespace is ignored.
 * For simplicity, overflow and underflow are not detected. If line can't
 * be read, returns FLT_MAX.
 */

/*
 * Reads a line of text from standard input and returns it as an
 * int in the range of [-2^31 + 1, 2^31 - 2], if possible; if text
 * does not represent such an int, user is prompted to retry. Leading
 * and trailing whitespace is ignored. For simplicity, overflow is not
 * detected. If line can't be read, returns INT_MAX.
 */

/*
 * Reads a line of text from standard input and returns an equivalent
 * long long in the range [-2^63 + 1, 2^63 - 2], if possible; if text
 * does not represent such a long long, user is prompted to retry.
 * Leading and trailing whitespace is ignored. For simplicity, overflow
 * is not detected. If line can't be read, returns LONGLONG_MAX.
 */

/*
 * Reads a line of text from standard input and returns it as a
 * string (char *), sans trailing newline character. (Ergo, if
 * user inputs only "\n", returns "" not NULL.) Returns NULL
 * upon error or no input whatsoever (i.e., just EOF). Leading
 * and trailing whitespace is not ignored. Stores string on heap
 * (via malloc); memory must be freed by caller to avoid leak.
 */
*/

string GetString(void);

#endif
# sigma1.c

* Computer Science 50
* David J. Malan

* Adds the numbers 1 through n.
* Demonstrates iteration.

---

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

// prototype
int sigma(int);

int main(void) {
    // ask user for a positive int
    int n;
    do {
        printf("Positive integer please: ");
        n = GetInt();
    } while (n < 1);

    // compute sum of 1 through n
    int answer = sigma(n);

    // report answer
    printf("%d\n", answer);
}

/*
* Returns sum of 1 through m; returns 0 if m is not positive.
*/

int sigma(int m) {
    // avoid risk of infinite loop
    if (m < 1)
return 0;

// return sum of 1 through m
int sum = 0;
for (int i = 1; i <= m; i++)
    sum += i;
return sum;
/* sigma2.c 

* Computer Science 50 
* David J. Malan 
* Adds the numbers 1 through n. 
* Demonstrates recursion. 
***************************************************************************/

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

// prototype 
int sigma(int); 

int main(void) { 
  // ask user for a positive int  
  int n;  
  do { 
    printf("Positive integer please: ");  
    n = GetInt(); 
  } while (n < 1); 
  // compute sum of 1 through n 
  int answer = sigma(n); 
  // report answer  
  printf("%d\n", answer); 
} 

/*  
* Returns sum of 1 through m; returns 0 if m is not positive.  
*/ 
int sigma(int m) { 
  // base case  
  if (m <= 0)
return 0;

else // recursive case
    return (m + sigma(m-1));
}
# string1.c

* Computer Science 50
* David J. Malan

* Prints a given string one character per line.
* Demonstrates strings as arrays of chars and use of strlen.

---

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

int main(void)
{
    // get line of text
    string s = GetString();

    // print string, one character per line
    if (s != NULL)
    {
        for (int i = 0; i < strlen(s); i++)
        {
            char c = s[i];
            printf("%c\n", c);
        }
    }
}
```
# string2.c

* Computer Science 50
* David J. Malan

* Prints a given string one character per line.
* Demonstrates strings as arrays of chars with slight optimization.

---

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

int main(void)
{
    // get line of text
    string s = GetString();
    // print string, one character per line
    if (s != NULL)
    {
        for (int i = 0, n = strlen(s); i < n; i++)
        {
            printf("%c\n", s[i]);
        }
    }
}
```

---

*prints a single line per character of the string.*