```c
/**
 * compare-1.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Compares two strings.
 *
 * Demonstrates strings as pointers to characters.
 */

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

int main(void)
{
    // get line of text
    printf("Say something: ");
    char* s = GetString();

    // get another line of text
    printf("Say something: ");
    char* t = GetString();

    // try to compare strings
    if (s != NULL && t != NULL)
    {
        if (strcmp(s, t) == 0)
        {
            printf("You typed the same thing!\n");
        }
        else
        {
            printf("You typed different things!\n");
        }
    }
}
```

```c
/**
 * copy-0.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Tries and fails to copy two strings.
 *
 * Demonstrates strings as pointers to arrays.
 */

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

int main(void)
{
    // get line of text
    printf("Say something: ");
    string s = GetString();
    if (s == NULL)
    {
        return 1;
    }

    // try (and fail) to copy string
    string t = s;

    // change "copy"
    printf("Capitalizing copy...\n");
    if (strlen(t) > 0)
    {
        t[0] = toupper(t[0]);
    }

    // print original and "copy"
    printf("Original: %s\n", s);
    printf("Copy:     %s\n", t);

    // success
    return 0;
}
```

```c
/**
 * copy-1.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Copies a string.
 *
 * Demonstrates strings as pointers to arrays.
 */

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

int main(void)
{
    // get line of text
    printf("Say something: ");
    char* s = GetString();
    if (s == NULL)
    {
        return 1;
    }

    // allocate enough space for copy
    char* t = malloc((strlen(s) + 1) * sizeof(char));
    if (t == NULL)
    {
        return 1;
    }

    // copy string, including '\0' at end
    for (int i = 0, n = strlen(s); i <= n; i++)
    {
        t[i] = s[i];
    }

    // change copy
    printf("Capitalizing copy...\n");
    if (strlen(t) > 0)
    {
        t[0] = toupper(t[0]);
    }

    // print original and copy
    printf("Original: %s\n", s);
```

```c
    printf("Copy:     %s\n", t);

    // free memory
    free(s);
    free(t);

    // success
    return 0;
}
```

```c
/**
 * copy-2.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Copies a string.
 *
 * Demonstrates pointer arithmetic.
 */

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

int main(void)
{
    // get line of text
    printf("Say something: ");
    char* s = GetString();
    if (s == NULL)
    {
        return 1;
    }

    // allocate enough space for copy
    char* t = malloc((strlen(s) + 1) * sizeof(char));
    if (t == NULL)
    {
        return 1;
    }

    // copy string, including '\0' at end
    for (int i = 0, n = strlen(s); i <= n; i++)
    {
        *(t + i) = *(s + i);
    }

    // change copy
    printf("Capitalizing copy...\n");
    if (strlen(t) > 0)
    {
        *t = toupper(*t);
    }

    // print original and copy
    printf("Original: %s\n", s);
```

```c
    printf("Copy:     %s\n", t);

    // success
    return 0;
}
```

```c
/**
 * pointers.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Prints a given string one character per line.
 *
 * Demonstrates pointer arithmetic.
 */

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

int main(void)
{
    // get line of text
    char* s = GetString();
    if (s == NULL)
    {
        return 1;
    }

    // print string, one character per line
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        printf("%c\n", *(s+i));
    }
}
```

```c
/**
 * structs-0.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Demonstrates use of structs.
 */

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

#include "structs.h"

// number of students
#define STUDENTS 3

int main(void)
{
    // declare students
    student students[STUDENTS];

    // populate students with user's input
    for (int i = 0; i < STUDENTS; i++)
    {
        printf("Student's name: ");
        students[i].name = GetString();

        printf("Student's dorm: ");
        students[i].dorm = GetString();
    }

    // now print students
    for (int i = 0; i < STUDENTS; i++)
    {
        printf("%s is in %s.\n", students[i].name, students[i].dorm);
    }

    // free memory
    for (int i = 0; i < STUDENTS; i++)
    {
        free(students[i].name);
        free(students[i].dorm);
    }
}
```

```c
/**
 * structs-1.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Demonstrates use of file I/O.
 */

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

#include "structs.h"

// number of students
#define STUDENTS 3

int main(void)
{
    // declare students
    student students[STUDENTS];

    // populate students with user's input
    for (int i = 0; i < STUDENTS; i++)
    {
        printf("Student's name: ");
        students[i].name = GetString();

        printf("Student's dorm: ");
        students[i].dorm = GetString();
    }

    // save students to disk
    FILE* file = fopen("students.csv", "w");
    if (file != NULL)
    {
        for (int i = 0; i < STUDENTS; i++)
        {
            fprintf(file, "%s,%s\n", students[i].name, students[i].dorm);
        }
        fclose(file);
    }

    // free memory
    for (int i = 0; i < STUDENTS; i++)
    {
```

```
49.         free(students[i].name);
50.         free(students[i].dorm);
51.     }
52. }
```

```c
/**
 * structs.h
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Defines a student.
 */

#include <cs50.h>

// structure representing a student
typedef struct
{
    string name;
    string dorm;
}
student;
```

```c
/**
 * swap.c
 *
 * David J. Malan
 * malan@harvard.edu
 *
 * Swaps two variables' values.
 *
 * Demonstrates passing by reference.
 */

#include <stdio.h>

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

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

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

/**
 * Swap arguments' values.
 */
void swap(int* a, int* b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```