compare0.c

```c
// Compares two strings' addresses

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

int main(void)
{
    // get two strings
    string s = get_string("s: ");
    string t = get_string("t: ");

    // compare strings' addresses
    if (s == t)
    {
        printf("same\n");
    }
    else
    {
        printf("different\n");
    }
}
```

compare1.c

```c
// Compares two strings for equality

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

int main(void)
{
    // get two strings
    char *s = get_string("s: ");
    char *t = get_string("t: ");

    // compare strings for equality
    if (strcmp(s, t) == 0)
    {
        printf("same\n");
    }
    else
    {
        printf("different\n");
    }
}
```

compare2.c

```c
// Compares two strings for equality while checking for errors

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

int main(void)
{
    // get a string
    char *s = get_string("s: ");
    if (s == NULL)
    {
        return 1;
    }

    // get another string
    char *t = get_string("t: ");
    if (t == NULL)
    {
        return 1;
    }

    // compare strings for equality
    if (strcmp(s, t) == 0)
    {
        printf("same\n");
    }
    else
    {
        printf("different\n");
    }
    return 0;
}
```

compare3.c

```c
// Compares two strings for equality while checking (succinctly) for errors

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

int main(void)
{
    // get a string
    char *s = get_string("s: ");
    if (!s)
    {
        return 1;
    }

    // get another string
    char *t = get_string("t: ");
    if (!t)
    {
        return 1;
    }

    // compare strings for equality
    if (strcmp(s, t) == 0)
    {
        printf("same\n");
    }
    else
    {
        printf("different\n");
    }
    return 0;
}
```

copy0.c

```c
// Capitalizes a string

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

int main(void)
{
    // get a string
    string s = get_string("s: ");

    // copy string's address
    string t = s;

    // capitalize first letter in string
    if (strlen(t) > 0)
    {
        t[0] = toupper(t[0]);
    }

    // print string twice
    printf("s: %s\n", s);
    printf("t: %s\n", t);
}
```

copy1.c

```c
// Capitalizes a copy of a string while checking for errors

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

int main(void)
{
    // get a string
    char *s = get_string("s: ");
    if (!s)
    {
        return 1;
    }

    // allocate memory for another string
    char *t = malloc((strlen(s) + 1) * sizeof(char));
    if (!t)
    {
        return 1;
    }

    // copy string into memory
    for (int i = 0, n = strlen(s); i <= n; i++)
    {
        t[i] = s[i];
    }

    // capitalize first letter in copy
    if (strlen(t) > 0)
    {
        t[0] = toupper(t[0]);
    }

    // print strings
    printf("s: %s\n", s);
    printf("t: %s\n", t);

    // free memory
    free(t);
    return 0;
}
```

memory.c

```c
// http://valgrind.org/docs/manual/quick-start.html#quick-start.prepare

#include <stdlib.h>

void f(void)
{
    int *x = malloc(10 * sizeof(int));
    x[10] = 0;
}

int main(void)
{
    f();
    return 0;
}
```

noswap.c

```c
// Fails to swap two integers

#include <stdio.h>

void swap(int a, int b);

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

    printf("x is %i, y is %i\n", x, y);
    swap(x, y);
    printf("x is %i, y is %i\n", x, y);
}

void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

scanf0.c

```c
// Gets an int from user using scanf

#include <stdio.h>

int main(void)
{
    int x;
    printf("x: ");
    scanf("%i", &x);
    printf("x: %i\n", x);
}
```

scanf1.c

```c
// Incorrectly gets a string from user using scanf

#include <stdio.h>

int main(void)
{
    char *s;
    printf("s: ");
    scanf("%s", s);
    printf("s: %s\n", s);
}
```

scanf2.c

```c
// Dangerously gets a string from user using scanf

#include <stdio.h>

int main(void)
{
    char s[5];
    printf("s: ");
    scanf("%s", s);
    printf("s: %s\n", s);
}
```

```c
// Prints a string's characters using square brackets

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

int main(void)
{
    // get a string
    char *s = get_string("string: ");
    if (!s)
    {
        return 1;
    }

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

string1.c

```c
// Prints a string's characters using pointer arithmetic

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

int main(void)
{
    // get a string
    char *s = get_string("string: ");
    if (!s)
    {
        return 1;
    }

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

struct0.c

```c
// Demonstrates structs

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

#include "struct.h"

int main(void)
{
    // allocate space for students
    int enrollment = get_int("enrollment: ");
    student students[enrollment];

    // prompt for students' names and dorms
    for (int i = 0; i < enrollment; i++)
    {
        students[i].name = get_string("name: ");
        students[i].dorm = get_string("dorm: ");
    }

    // print students' names and dorms
    for (int i = 0; i < enrollment; i++)
    {
        printf("%s is in %s.\n", students[i].name, students[i].dorm);
    }
}
```

```c
// Demonstrates file I/O

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

#include "structs.h"

int main(void)
{
    // allocate memory for students
    int enrollment = get_int("enrollment: ");
    student students[enrollment];

    // prompt for students' names and dorms
    for (int i = 0; i < enrollment; i++)
    {
        students[i].name = get_string("name: ");
        students[i].dorm = get_string("dorm: ");
    }

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

struct.h

```c
// Represents a student

typedef struct
{
    char *name;
    char *dorm;
}
student;
```

swap.c

```c
// Swaps two integers using pointers

#include <stdio.h>

void swap(int *a, int *b);

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

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

void swap(int *a, int *b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```