

```
1. /**
2.  * compare-1.c
3.  *
4.  * David J. Malan
5.  * malan@harvard.edu
6.  *
7.  * Compares two strings.
8.  *
9.  * Demonstrates strings as pointers to characters.
10. */
11.
12. #include <cs50.h>
13. #include <stdio.h>
14. #include <string.h>
15.
16. int main(void)
17. {
18.     // get line of text
19.     printf("Say something: ");
20.     char* s = GetString();
21.
22.     // get another line of text
23.     printf("Say something: ");
24.     char* t = GetString();
25.
26.     // try to compare strings
27.     if (s != NULL && t != NULL)
28.     {
29.         if (strcmp(s, t) == 0)
30.         {
31.             printf("You typed the same thing!\n");
32.         }
33.         else
34.         {
35.             printf("You typed different things!\n");
36.         }
37.     }
38. }
```

```
1. /**
2.  * copy-0.c
3.  *
4.  * David J. Malan
5.  * malan@harvard.edu
6.  *
7.  * Tries and fails to copy two strings.
8.  *
9.  * Demonstrates strings as pointers to arrays.
10. */
11.
12. #include <cs50.h>
13. #include <ctype.h>
14. #include <stdio.h>
15. #include <string.h>
16.
17. int main(void)
18. {
19.     // get line of text
20.     printf("Say something: ");
21.     string s = GetString();
22.     if (s == NULL)
23.     {
24.         return 1;
25.     }
26.
27.     // try (and fail) to copy string
28.     string t = s;
29.
30.     // change "copy"
31.     printf("Capitalizing copy...\n");
32.     if (strlen(t) > 0)
33.     {
34.         t[0] = toupper(t[0]);
35.     }
36.
37.     // print original and "copy"
38.     printf("Original: %s\n", s);
39.     printf("Copy:      %s\n", t);
40.
41.     // success
42.     return 0;
43. }
```

```
1. /**
2.  * copy-1.c
3.  *
4.  * David J. Malan
5.  * malan@harvard.edu
6.  *
7.  * Copies a string.
8.  *
9.  * Demonstrates strings as pointers to arrays.
10. */
11.
12. #include <cs50.h>
13. #include <ctype.h>
14. #include <stdio.h>
15. #include <string.h>
16.
17. int main(void)
18. {
19.     // get line of text
20.     printf("Say something: ");
21.     char* s = GetString();
22.     if (s == NULL)
23.     {
24.         return 1;
25.     }
26.
27.     // allocate enough space for copy
28.     char* t = malloc((strlen(s) + 1) * sizeof(char));
29.     if (t == NULL)
30.     {
31.         return 1;
32.     }
33.
34.     // copy string, including '\0' at end
35.     for (int i = 0, n = strlen(s); i <= n; i++)
36.     {
37.         t[i] = s[i];
38.     }
39.
40.     // change copy
41.     printf("Capitalizing copy...\n");
42.     if (strlen(t) > 0)
43.     {
44.         t[0] = toupper(t[0]);
45.     }
46.
47.     // print original and copy
48.     printf("Original: %s\n", s);
```

```
49.     printf("Copy:   %s\n", t);
50.
51.     // free memory
52.     free(s);
53.     free(t);
54.
55.     // success
56.     return 0;
57. }
```

```
1. /**
2.  * copy-2.c
3.  *
4.  * David J. Malan
5.  * malan@harvard.edu
6.  *
7.  * Copies a string.
8.  *
9.  * Demonstrates pointer arithmetic.
10. */
11.
12. #include <cs50.h>
13. #include <ctype.h>
14. #include <stdio.h>
15. #include <string.h>
16.
17. int main(void)
18. {
19.     // get line of text
20.     printf("Say something: ");
21.     char* s = GetString();
22.     if (s == NULL)
23.     {
24.         return 1;
25.     }
26.
27.     // allocate enough space for copy
28.     char* t = malloc((strlen(s) + 1) * sizeof(char));
29.     if (t == NULL)
30.     {
31.         return 1;
32.     }
33.
34.     // copy string, including '\0' at end
35.     for (int i = 0, n = strlen(s); i <= n; i++)
36.     {
37.         *(t + i) = *(s + i);
38.     }
39.
40.     // change copy
41.     printf("Capitalizing copy...\n");
42.     if (strlen(t) > 0)
43.     {
44.         *t = toupper(*t);
45.     }
46.
47.     // print original and copy
48.     printf("Original: %s\n", s);
```

```
49.     printf("Copy:   %s\n", t);
50.
51.     // success
52.     return 0;
53. }
```

```
1. /**
2.  * pointers.c
3.  *
4.  * David J. Malan
5.  * malan@harvard.edu
6.  *
7.  * Prints a given string one character per line.
8.  *
9.  * Demonstrates pointer arithmetic.
10. */
11.
12. #include <cs50.h>
13. #include <stdio.h>
14. #include <string.h>
15.
16. int main(void)
17. {
18.     // get line of text
19.     char* s = GetString();
20.     if (s == NULL)
21.     {
22.         return 1;
23.     }
24.
25.     // print string, one character per line
26.     for (int i = 0, n = strlen(s); i < n; i++)
27.     {
28.         printf("%c\n", *(s+i));
29.     }
30. }
```

```
1. /**
2.  * structs-0.c
3.  *
4.  * David J. Malan
5.  * malan@harvard.edu
6.  *
7.  * Demonstrates use of structs.
8.  */
9.
10. #include <cs50.h>
11. #include <stdio.h>
12. #include <string.h>
13.
14. #include "structs.h"
15.
16. // number of students
17. #define STUDENTS 3
18.
19. int main(void)
20. {
21.     // declare students
22.     student students[STUDENTS];
23.
24.     // populate students with user's input
25.     for (int i = 0; i < STUDENTS; i++)
26.     {
27.         printf("Student's name: ");
28.         students[i].name = GetString();
29.
30.         printf("Student's dorm: ");
31.         students[i].dorm = GetString();
32.     }
33.
34.     // now print students
35.     for (int i = 0; i < STUDENTS; i++)
36.     {
37.         printf("%s is in %s.\n", students[i].name, students[i].dorm);
38.     }
39.
40.     // free memory
41.     for (int i = 0; i < STUDENTS; i++)
42.     {
43.         free(students[i].name);
44.         free(students[i].dorm);
45.     }
46. }
```



```
1. /**
2.  * structs-1.c
3.  *
4.  * David J. Malan
5.  * malan@harvard.edu
6.  *
7.  * Demonstrates use of file I/O.
8.  */
9.
10. #include <cs50.h>
11. #include <stdio.h>
12. #include <stdlib.h>
13. #include <string.h>
14.
15. #include "structs.h"
16.
17. // number of students
18. #define STUDENTS 3
19.
20. int main(void)
21. {
22.     // declare students
23.     student students[STUDENTS];
24.
25.     // populate students with user's input
26.     for (int i = 0; i < STUDENTS; i++)
27.     {
28.         printf("Student's name: ");
29.         students[i].name = GetString();
30.
31.         printf("Student's dorm: ");
32.         students[i].dorm = GetString();
33.     }
34.
35.     // save students to disk
36.     FILE* file = fopen("students.csv", "w");
37.     if (file != NULL)
38.     {
39.         for (int i = 0; i < STUDENTS; i++)
40.         {
41.             fprintf(file, "%s,%s\n", students[i].name, students[i].dorm);
42.         }
43.         fclose(file);
44.     }
45.
46.     // free memory
47.     for (int i = 0; i < STUDENTS; i++)
48.     {
```

---

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

```
1. /**
2.  * structs.h
3.  *
4.  * David J. Malan
5.  * malan@harvard.edu
6.  *
7.  * Defines a student.
8.  */
9.
10. #include <cs50.h>
11.
12. // structure representing a student
13. typedef struct
14. {
15.     string name;
16.     string dorm;
17. }
18. student;
```

```
1. /**
2.  * swap.c
3.  *
4.  * David J. Malan
5.  * malan@harvard.edu
6.  *
7.  * Swaps two variables' values.
8.  *
9.  * Demonstrates passing by reference.
10. */
11.
12. #include <stdio.h>
13.
14. // function prototype
15. void swap(int* a, int* b);
16.
17. int main(void)
18. {
19.     int x = 1;
20.     int y = 2;
21.
22.     printf("x is %i\n", x);
23.     printf("y is %i\n", y);
24.     printf("Swapping...\n");
25.     swap(&x, &y);
26.     printf("Swapped!\n");
27.     printf("x is %i\n", x);
28.     printf("y is %i\n", y);
29. }
30.
31. /**
32.  * Swap arguments' values.
33.  */
34. void swap(int* a, int* b)
35. {
36.     int tmp = *a;
37.     *a = *b;
38.     *b = tmp;
39. }
```