Notes

Introduction to Computer Science (CS50) on EdX

Sparsh Jain

November 26, 2020

Contents

List of Programs

Chapter 1

Computational Thinking, Scratch

1.1 Binary Number System

1.2 Algorithms

1.3 Time Complexity

1.4 Pseudocode

1.5 Scratch

This was only an introductory lecture. Click here for more details.

Chapter 2

C

2.1 Hello World

```c
#include <stdio.h>

int main(void)
{
    printf("Hello, World!\n");
}
```

Program 2.1: Hello World in C

Remark. Need to compile using a compiler like `clang` or `gcc`.

2.2 Input

Remark. In case of errors in compiling, start by trying to *fix* the first one, and so on.

Remark. Use `-lcs50` to link `cs50.h` header.

Remark. Use `make` to ease your life compiling!

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

int main(void)
{
    string answer = get_string("What's your name?\n");
    printf("Hello, %s!\n", answer);
}
```

Program 2.2: Hello User in C

2.3 Initialization

```
int counter = 0;
```

2.4 Increment

```
counter = counter + 1;
counter += 1;
counter++; // Syntactic Sugar
```

2.5 Conditionals

```
if (x < y)
{
        printf("x is less than y!\n");
}
else if (x > y)
{
        printf("x is greater than y!\n");
}
else // if (x == y)
{
        printf("x is equal to y!\n");
}
```

2.6 Loops

2.6.1 While Loop

Infinite Loop

```
while(true)
{

}
```

Repeat

```
int i = 0;
while(i < 50)
```

```
        {
                printf("Hello World!\n");
                i = i+1;
        }
```

2.6.2 For Loop

```
        for(int i = 0; i < 50; i += 1)
        {
                printf("Hello World!\n");
        }
```

2.7 Additional Info

2.7.1 Datatypes

Some of these (like `string`) are implemented in `cs50.h` library.

- `bool`
- `char`
- `double`
- `float`
- `int`
- `long`
- `string`
- ...

2.7.2 Functions

They are implemented in `cs50.h` library.

- `get_char`
- `get_float`
- `get_double`

- `get_int`
- `get_long`
- `get_string`
- ...

2.7.3 Placeholders

- `%c` for `char`
- `%f` for `float`
- `%i` for `int`
- `%li` for `long`
- `%s` for `string`

2.7.4 Arithmetic Operations

- `+`
- `-`
- `*`
- `/`
- `%`

2.8 Examples

2.8.1 Arithmetic

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

int main(void)
{
    int age = get_int("What's your age?\n");
    // int days = age * 365;
    // printf("You are atleast %i days old.\n", days);
    printf("You are atleast %i days old.\n", age * 365);
}
```

Program 2.3: int.c

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

int main(void)
{
    float price = get_float("What's the price?\n");
    // printf("Your total is %f.\n", price * 1.18);
    printf("Your total is %.2f.\n", price * 1.18);
}
```

Program 2.4: float.c

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

int main(void)
{
    int n = get_int("n: ");

    if (n % 2 == 0)
    {
        printf("even.\n");
    }
    else
    {
        printf("odd.\n");
    }
}
```

Program 2.5: parity.c

2.8.2 Conditional

```c
// Conditions and relational operators

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

int main(void)
{
    // Prompt user for x
    int x = get_int("x: ");

    // Prompt user for y
    int y = get_int("y: ");

    // Compare x and y
    if (x < y)
    {
        printf("x is less than y\n");
    }
    else if (x > y)
    {
        printf("x is greater than y\n");
    }
    else
    {
        printf("x is equal to y\n");
    }
}
```

Program 2.6: conditions.c

2.8.3 Logical

```c
// Logical operators
#include <cs50.h>
#include <stdio.h>
int main(void)
{
    // Prompt user to agree
    char c = get_char("Do you agree?\n");
    // Check whether agreed
    if (c == 'Y'  c == 'y')
    {
        printf("Agreed.\n");
    }
    else if (c == 'N'  c == 'n')
    {
        printf("Not agreed.\n");
    }
}
```

Program 2.7: agree.c

2.8.4 Loop

```c
// Opportunity for better design

#include <stdio.h>

int main(void)
{
    printf("cough\n");
    printf("cough\n");
    printf("cough\n");
}
```

Program 2.8: cough0.c

```c
// Better design

#include <stdio.h>

int main(void)
{
    for (int i = 0; i < 3; i++)
    {
        printf("cough\n");
    }
}
```

Program 2.9: cough1.c

2.8.5 Function

```c
// Abstraction

#include <stdio.h>

void cough(void);

int main(void)
{
    for (int i = 0; i < 3; i++)
    {
        cough();
    }
}

// Cough once
void cough(void)
{
    printf("cough\n");
}
```

Program 2.10: cough2.c

```c
// Abstraction with parameterization

#include <stdio.h>

void cough(int n);

int main(void)
{
    cough(3);
}

// Cough some number of times
void cough(int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("cough\n");
    }
}
```

Program 2.11: cough3.c

```c
// Abstraction and scope

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

int get_positive_int(void);

int main(void)
{
    int i = get_positive_int();
    printf("%i\n", i);
}

// Prompt user for positive integer
int get_positive_int(void)
{
    int n;
    do
    {
        n = get_int("Positive Integer: ");
    }
    while (n < 1);
    return n;
}
```

Program 2.12: positive.c

```c
// Prints a row of 4 question marks

#include <stdio.h>

int main(void)
{
    printf("????\n");
}
```

Program 2.13: mario0.c

```c
// Prints a row of n question marks with a loop

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

int main(void)
{
    int n;
    do
    {
        n = get_int("Width: ");
    }
    while (n < 1);
    for (int i = 0; i < n; i++)
    {
        printf("?");
    }
    printf("\n");
}
```

Program 2.14: mario2.c

```c
// Prints an n-by-n grid of bricks with a loop

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

int main(void)
{
    int n;
    do
    {
        n = get_int("Size: ");
    }
    while (n < 1);
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("#");
        }
        printf("\n");
    }
}
```

Program 2.15: mario8.c

2.9 Limitations

```c
// Floating-point arithmetic with float

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

int main(void)
{
    // Prompt user for x
    float x = get_float("x: ");

    // Prompt user for y
    float y = get_float("y: ");

    // Perform division
    printf("x / y = %.50f\n", x / y);
}
```

Program 2.16: floats.c

```c
// Integer overflow

#include <stdio.h>
#include <unistd.h>

int main(void)
{
    // Iteratively double i
    for (int i = 1; ; i *= 2)
    {
        printf("%i\n", i);
        sleep(1);
    }
}
```

Program 2.17: overflow.c

Click here for more examples.

Chapter 3

Arrays

3.1 Compiling

3.1.1 Preprocessing

Expansion/Inclusion of header files, macros, etc.

3.1.2 Compiling

C code → Assembly code.

3.1.3 Assembling

Assembly code → Machine code.

3.1.4 Linking

Linking all relevent files.

3.2 Debugging

- Can use `help50` to understand error msgs in this course.
- Can use (poor man's) `printf`.
- Can use `debug50` for proper debugging (in this course).

Remark. Use `style50` for styling your code.

3.3 Casting

```c
// Prints ASCII codes

#include <stdio.h>

int main(void)
{
    char c1 = 'H';
    char c2 = 'I';
    char c3 = '!';
    printf("%i %i %i\n", c1, c2, c3);
}
```

Program 3.1: casting

3.4 Array

Follow through the following examples:

```c
// Averages three numbers

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

int main(void)
{
    // Scores
    int score1 = 72;
    int score2 = 73;
    int score3 = 33;

    // Print average
    printf("Average: %i\n", (score1 + score2 + score3) / 3);
}
```

Program 3.2: scores0.c

```c
// Averages three numbers using an array

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

int main(void)
{
    // Scores
    int scores[3];
    scores[0] = 72;
    scores[1] = 73;
    scores[2] = 33;

    // Print average
    printf("Average: %i\n", (scores[0] + scores[1] + scores[2])
     ↪  / 3);
}
```

Program 3.3: scores1.c

3.5 String

`string` is just (or a little more) than an array of `char`s.

```c
// Averages three numbers using an array and a constant

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

const int N = 3;

int main(void)
{
    // Scores
    int scores[N];
    scores[0] = 72;
    scores[1] = 73;
    scores[2] = 33;

    // Print average
    printf("Average: %i\n", (scores[0] + scores[1] + scores[2])
     ↪  / N);
}
```

Program 3.4: scores2.c

```c
// Averages numbers using a helper function

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

float average(int length, int array[]);

int main(void)
{
    // Get number of scores
    int n = get_int("Scores:  ");

    // Get scores
    int scores[n];
    for (int i = 0; i < n; i++)
    {
        scores[i] = get_int("Score %i: ", i + 1);
    }

    // Print average
    printf("Average: %.1f\n", average(n, scores));
}

float average(int length, int array[])
{
    int sum = 0;
    for (int i = 0; i < length; i++)
    {
        sum += array[i];
    }
    return (float) sum / (float) length;
}
```

Program 3.5: scores3.c

```c
// Stores names using an array

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

int main(void)
{
    // Names
    string names[4];
    names[0] = "EMMA";
    names[1] = "RODRIGO";
    names[2] = "BRIAN";
    names[3] = "DAVID";

    // Print Emma's name
    printf("%s\n", names[0]);
    printf("%c%c%c%c\n", names[0][0], names[0][1], names[0][2],
     ↪  names[0][3]);
}
```

Program 3.6: names.c

```c
// Prints string char by char, one per line

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

int main(void)
{
    string s = get_string("Input:  ");
    printf("Output: ");
    for (int i = 0; s[i] != '\0'; i++)
    {
        printf("%c", s[i]);
    }
    printf("\n");
}
```

Program 3.7: string0.c

```c
// Prints string char by char, one per line, using strlen

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

int main(void)
{
    string s = get_string("Input:  ");
    printf("Output: ");
    for (int i = 0; i < strlen(s); i++)
    {
        printf("%c", s[i]);
    }
    printf("\n");
}
```

Program 3.8: string1.c

```c
// Prints string char by char, one per line, using strlen,
    remembering string's length

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

int main(void)
{
    string s = get_string("Input: ");
    printf("Output: ");
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        printf("%c", s[i]);
    }
    printf("\n");
}
```

Program 3.9: string2.c

```c
// Uppercases a string

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

int main(void)
{
    string s = get_string("Before: ");
    printf("After:  ");
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        if (s[i] >= 'a' && s[i] <= 'z')
        {
            printf("%c", s[i] - 32);
        }
        else
        {
            printf("%c", s[i]);
        }
    }
    printf("\n");
}
```

Program 3.10: uppercase0.c

```c
// Uppercases string using ctype library (and an unnecessary
  ↪  condition)

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

int main(void)
{
    string s = get_string("Before: ");
    printf("After:  ");
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        if (islower(s[i]))
        {
            printf("%c", toupper(s[i]));
        }
        else
        {
            printf("%c", s[i]);
        }
    }
    printf("\n");
}
```

Program 3.11: uppercase1.c

3.6 Command Line Arguments

```c
// Printing a command-line argument

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

int main(int argc, string argv[])
{
    if (argc == 2)
    {
        printf("hello, %s\n", argv[1]);
    }
    else
    {
        printf("hello, world\n");
    }
}
```

Program 3.12: argv.c

```c
// Printing characters in an array of strings

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

int main(int argc, string argv[])
{
    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");
    }
}
```

Program 3.13: argv2.c

```c
// Returns explicit value from main

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

int main(int argc, string argv[])
{
    if (argc != 2)
    {
        printf("missing command-line argument\n");
        return 1;
    }
    printf("hello, %s\n", argv[1]);
    return 0;
}
```

Program 3.14: exit.c

Chapter 4

Algorithms

4.1 Linear Search

```
for i from 0 to n-1
        if ith element is 50
                return true;
return false;
```

Program 4.1: Linear Search Pseudocode

4.2 Binary Search

```
if no items
        return false;
if middle item is 50
        return true;
else if 50 < middle item
        search left half
else if 50 > middle item
        search right half
```

Program 4.2: Binary Search Pseudocode

4.3 Efficiency

4.3.1 $\mathcal{O}$ Notation:

Worst case scenario

$$\begin{aligned}
n^2 &: \mathcal{O}(n^2) \\
n \log_n n &: \mathcal{O}(n \log n) \\
n &: \mathcal{O}(n) \; (LinearSearch) \\
n/2 &: \mathcal{O}(n) \\
\log_2 n &: \mathcal{O}(\log n) \; (BinarySearch) \\
constant &: \mathcal{O}(1)
\end{aligned}$$

4.3.2 Ω Notation:

Best case scenario

$$\begin{aligned}
&\Omega(n^2) \\
&\Omega(n \log n) \\
&\Omega(n) \\
&\Omega(n) \\
&\Omega(\log n) \\
&\Omega(1)
\end{aligned}$$

Q: Better to have a really good $\mathcal{O}$ value or a really good Ω value?

A: $\mathcal{O}$, or even *average* case.

4.4 Examples

4.4.1 Linear Search

Numbers

```c
// Implements linear search for numbers

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

int main(void)
{
    // An array of numbers
    int numbers[] = {4, 8, 15, 16, 23, 42};

    // Search for 50
    for (int i = 0; i < 6; i++)
    {
        if (numbers[i] == 50)
        {
            printf("Found\n");
            return 0;
        }
    }
    printf("Not found\n");
    return 1;
}
```

Program 4.3: Linear Search on numbers

Names

```c
// Implements linear search for names

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

int main(void)
{
    // An array of names
    string names[] = {"EMMA", "RODRIGO", "BRIAN", "DAVID"};

    // Search for EMMA
    for (int i = 0; i < 4; i++)
    {
        if (strcmp(names[i], "EMMA") == 0)
        {
            printf("Found\n");
            return 0;
        }
    }
    printf("Not found\n");
    return 1;
}
```

Program 4.4: Linear Search on names

4.4.2 Bad Design

Correct/Working code but bad design!

```c
// Implements a phone book without structs

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

int main(void)
{
    string names[] = {"EMMA", "RODRIGO", "BRIAN", "DAVID"};
    string numbers[] = {"617-555-0100", "617-555-0101",
    ↪ "617-555-0102", "617-555-0103"};

    for (int i = 0; i < 4; i++)
    {
        if (!strcmp(names[i], "EMMA"))
        {
            printf("Found %s\n", numbers[i]);
            return 0;
        }
    }
    printf("Not found\n");
    return 1;
}
```

Program 4.5: Linear Search in a phonebook

4.4.3 Good Design - `typedef struct`

Using `typedef struct` for better design!

```
// Implements a phone book with structs

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

typedef struct
{
    string name;
    string number;
}
person;

int main(void)
{
    person people[4];

    people[0].name = "EMMA";
    people[0].number = "617-555-0100";

    people[1].name = "RODRIGO";
    people[1].number = "617-555-0101";

    people[2].name = "BRIAN";
    people[2].number = "617-555-0102";

    people[3].name = "DAVID";
    people[3].number = "617-555-0103";

    // Search for EMMA
```

```c
    for (int i = 0; i < 4; i++)
    {
        if (strcmp(people[i].name, "EMMA") == 0)
        {
            printf("Found %s\n", people[i].number);
            return 0;
        }
    }
    printf("Not found\n");
    return 1;
}
```

Program 4.6: Linear Search in phonebook with `typedef struct`

4.5 Bubble Sort

```
repeat n-1 times
        for i = 0 to n-2
                if ith and i+1th elements out of order
                        swap them
```

$\mathcal{O}(n^2)$

$\Omega(n^2)$

4.6 Selection Sort

```
for i from 0 to n-1
        find smallest item between ith item and last item
        swap smallest item and ith item
```

$\mathcal{O}(n^2)$

$\Omega(n^2)$

4.7 Better Bubble Sort

```
repeat until swap
        for i = 0 to n-2
                if ith and i+1th elements out of order
                        swap them
```

$\mathcal{O}(n^2)$

$\Omega(n)$

4.8 Recursion

```
1  Pick up phone book
2  Open to middle of phone book
3  Look at page
4  if Smith is on page
5          Call Mike
6  else if Smith is earlier in book
7          Open to middle of left half of book
8          Go back to line 3
9  else if Smith is later in book
10         Open to middle of right half of book
11         Go back to line 3
12 else
13         Quit
```

Program 4.7: Iteration Pseudocode

Can we do a better design?

```
Pick up phone book
Open to middle of phone book
Look at page
if Smith is on page
        Call Mike
else if Smith is earlier in book
        Search left half of book
else if Smith is later in book
        Search right half of book
```

```
else
        Quit
```

Program 4.8: Recursion Pseudocode

```c
// Draws a pyramid using iteration

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

void draw(int h);

int main(void)
{
    // Get height of pyramid
    int height = get_int("Height: ");

    // Draw pyramid
    draw(height);
}

void draw(int h)
{
    // Draw pyramid of height h
    for (int i = 1; i <= h; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            printf("#");
        }
        printf("\n");
    }
}
```

Program 4.9: Iteration C code

```c
// Draws a pyramid using recursion

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

```c

void draw(int h);

int main(void)
{
    // Get height of pyramid
    int height = get_int("Height: ");

    // Draw pyramid
    draw(height);
}

void draw(int h)
{
    // If nothing to draw
    if (h == 0)
    {
        return;
    }

    // Draw pyramid of height h - 1
    draw(h - 1);

    // Draw one more row of width h
    for (int i = 0; i < h; i++)
    {
        printf("#");
    }
    printf("\n");
}
```

Program 4.10: Recursion C code

4.9 Merge Sort

```
if only 1 item
        return
else
        sort left half of items
        sort right half of items
        merge sorted halves
```

Program 4.11: Merge Sort Pseudocode

$\mathcal{O}(n \log n)$

$\Omega(n \log n)$

4.9.1 Θ Notation

When $\mathcal{O} = \Omega$!

Chapter 5

Memory

Removing the training wheels `#include <cs50.h>` from now!

5.1 Hexadecimal

Digits: $\{1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$

Ambiguity: Prefix the number with $0x$

5.2 Addresses

```c
// Prints an integer

#include <stdio.h>

int main(void)
{
    int n = 50;
    printf("%i\n", n);
}
```

Program 5.1: integer

```c
// Prints an integer's address

#include <stdio.h>

int main(void)
{
    int n = 50;
    printf("%p\n", &n);
}
```

Program 5.2: address of an integer

```c
// Prints an integer via its address

#include <stdio.h>

int main(void)
{
    int n = 50;
    printf("%i\n", *&n);
}
```

Program 5.3: address2.c

5.2.1 Operators

$$\& = \text{Get the address}$$
$$* = \text{Go to the address}$$

5.3 Pointers

```c
// Stores and prints an integer's address

#include <stdio.h>

int main(void)
{
    int n = 50;
    int *p = &n;
    printf("%p\n", p);
}
```

Program 5.4: accessing an address

```c
// Stores and prints an integer via its address

#include <stdio.h>

int main(void)
{
    int n = 50;
    int *p = &n;
    printf("%i\n", *p);
}
```

Program 5.5: pointers

5.4 Strings

There are no strings. Strings are just pointers.

```c
// Prints a string

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

int main(void)
{
    string s = "EMMA";
    printf("%s\n", s);
}
```

Program 5.6: strings

```c
// Prints a string's address

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

int main(void)
{
    string s = "EMMA";
    printf("%p\n", s);
}
```

Program 5.7: strings are pointers

```c
// Prints a string's address as well the addresses of its
↪  chars

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

int main(void)
{
    string s = "EMMA";
    printf("%p\n", s);
    printf("%p\n", &s[0]);
    printf("%p\n", &s[1]);
    printf("%p\n", &s[2]);
    printf("%p\n", &s[3]);
    printf("%p\n", &s[4]);
}
```

Program 5.8: strings are `char[]`
addresses are consecutive in arrays

```
// Prints a string's chars

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

int main(void)
{
    string s = "EMMA";
    printf("%c\n", s[0]);
    printf("%c\n", s[1]);
    printf("%c\n", s[2]);
    printf("%c\n", s[3]);
}
```

Program 5.9: accessing characters in a string

```
// Stores and prints a string's address via pointer arithmetic

#include <stdio.h>

int main(void)
{
    char *s = "EMMA";
    printf("%c\n", *s);
    printf("%c\n", *(s+1));
    printf("%c\n", *(s+2));
    printf("%c\n", *(s+3));
}
```

Program 5.10: accessing characters in a `char *`

5.5 String Comparision

```c
// Compares two integers

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

int main(void)
{
    // Get two integers
    int i = get_int("i: ");
    int j = get_int("j: ");

    // Compare integers
    if (i == j)
    {
        printf("Same\n");
    }
    else
    {
        printf("Different\n");
    }
}
```

Program 5.11: comparing integers

```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");
    }
}
```

Program 5.12: attempting to compare strings directly

```c
// Compares two strings using strcmp

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

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

    // Compare strings
    if (strcmp(s, t) == 0)
    {
        printf("Same\n");
    }
    else
    {
        printf("Different\n");
    }
}
```

Program 5.13: comparing strings properly

5.6 String Copy

```
// 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);
}
```

Program 5.14: attempting to copying strings directly

```
// Capitalizes a copy of a string

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

int main(void)
{
    // Get a string
    char *s = get_string("s: ");

    // Allocate memory for another string
    char *t = malloc(strlen(s) + 1);

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

    // Capitalize copy
    t[0] = toupper(t[0]);

    // Print strings
    printf("s: %s\n", s);
    printf("t: %s\n", t);
}
```

Program 5.15: copy strings properly

Just use `strcpy(target, source)` to copy strings.

5.7 Malloc and Free

`malloc`: Allocate Memory and return its address.

`free`: Free Memory (prevent leaking).

5.8 Buffer Overflow

```
// 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;
```

```c
}

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

Program 5.16: buffer overflow

5.9 Swap

Pass by *value* vs pass by *reference*

```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;
}
```

Program 5.17: naive attempt at swap

```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;
}
```

Program 5.18: swap

5.10 scanf

```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);
}
```

Program 5.19: scanning an integer

```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);
}
```

Program 5.20: scanning a string in unintialized

```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);
}
```

Program 5.21: scanning a long string in small array

5.11 File I/O

```c
// Saves names and numbers to a CSV file

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

int main(void)
{
    // Open CSV file
    FILE *file = fopen("phonebook.csv", "a");
    if (!file)
    {
        return 1;
    }

    // Get name and number
    string name = get_string("Name: ");
    string number = get_string("Number: ");

    // Print to file
    fprintf(file, "%s,%s\n", name, number);

    // Close file
    fclose(file);
}
```

Program 5.22: files in c

```
Sparsh,6238-098-518
```

Program 5.23: phonebook.csv

```c
// Detects if a file is a JPEG

#include <stdio.h>

int main(int argc, char *argv[])
{
    // Check usage
    if (argc != 2)
    {
        return 1;
    }

    // Open file
    FILE *file = fopen(argv[1], "r");
    if (!file)
    {
        return 1;
    }

    // Read first three bytes
    unsigned char bytes[3];
    fread(bytes, 3, 1, file);

    // Check first three bytes
    if (bytes[0] == 0xff && bytes[1] == 0xd8 && bytes[2] ==
     ↪  0xff)
    {
        printf("Maybe\n");
    }
    else
    {
        printf("No\n");
    }

    // Close file
    fclose(file);
}
```

Program 5.24: check jpeg or not

Chapter 6

Data Structures

6.1 Arrays

- Fixed size
- Resizing ≡ Relocating
- This implies insert = $\mathcal{O}(n)$
- Search = $\mathcal{O}(\log n)$

```c
// Implements a list of numbers with an array of fixed size

#include <stdio.h>

int main(void)
{
    // List of size 3
    int list[3];

    // Initialize list with numbers
    list[0] = 1;
    list[1] = 2;
    list[2] = 3;

    // Print list
    for (int i = 0; i < 3; i++)
    {
        printf("%i\n", list[i]);
    }
}
```

Program 6.1: array with hardcoded size

```
// Implements a list of numbers with an array of dynamic size
//
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    // List of size 3
    int *list = malloc(3 * sizeof(int));
    if (list == NULL)
    {
        return 1;
    }

    // Initialize list of size 3 with numbers
    list[0] = 1;
    list[1] = 2;
    list[2] = 3;

    // List of size 4
    int *tmp = malloc(4 * sizeof(int));
    if (tmp == NULL)
    {
        return 1;
    }

    // Copy list of size 3 into list of size 4
    for (int i = 0; i < 3; i++)
    {
        tmp[i] = list[i];
    }

    // Add number to list of size 4
    tmp[3] = 4;

    // Free list of size 3
```

```c
    free(list);

    // Remember list of size 4
    list = tmp;

    // Print list
    for (int i = 0; i < 4; i++)
    {
        printf("%i\n", list[i]);
    }

    // Free list
    free(list);
}
```

Program 6.2: array with dynamic size using `malloc`

```c
// Implements a list of numbers with an array of dynamic size
 ↪   using realloc

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

int main(void)
{
    // List of size 3
    int *list = malloc(3 * sizeof(int));
    if (list == NULL)
    {
        return 1;
    }

    // Initialize list of size 3 with numbers
    list[0] = 1;
    list[1] = 2;
    list[2] = 3;

    // Resize list to be of size 4
    int *tmp = realloc(list, 4 * sizeof(int));
    if (tmp == NULL)
    {
        return 1;
    }
    list = tmp;

    // Add number to list
    list[3] = 4;

    // Print list
    for (int i = 0; i < 4; i++)
    {
        printf("%i\n", list[i]);
    }

    // Free list
    free(list);
}
```

Program 6.3: array with dynamic size using `realloc`

6.2 Data Structures

Structures to store data. In *c*, it basically revolves around

- `struct`
- `.`
- `*`

6.3 Linked List

```c
// Implements a list of numbers with linked list

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

// Represents a node
typedef struct node
{
    int number;
    struct node *next;
}
node;

int main(void)
{
    // List of size 0
    node *list = NULL;

    // Add number to list
    node *n = malloc(sizeof(node));
    if (n == NULL)
    {
        return 1;
    }
    n->number = 1;
    n->next = NULL;
    list = n;

    // Add number to list
    n = malloc(sizeof(node));
    if (n == NULL)
    {
        return 1;
    }
    n->number = 2;
    n->next = NULL;
    list->next = n;

    // Add number to list
    n = malloc(sizeof(node));
    if (n == NULL)
    {
        return 1;
    }
    n->number = 3;
    n->next = NULL;
    list->next->next = n;

    // Print list
    for (node *tmp = list; tmp != NULL; tmp = tmp->next)
    {
        printf("%i\n", tmp->number);
    }

    // Free list
    while (list != NULL)
    {
        node *tmp = list->next;
        free(list);
        list = tmp;
    }
}
```

Program 6.4: linked list

We have now lost random access. So:

- Search = $\mathcal{O}(n)$
- Insert = $\mathcal{O}(n)$

6.4 Tree

Think of as multi-dimensional linked lists.

6.4.1 Binary Search Tree

```
typedef struct node
{
        int number;
        struct node *left;
        struct node *right;
}
node;
```

Program 6.5: node for a binary tree

```c
bool search(node *tree, int n)
{
        if (tree == NULL)
        {
                return false;
        }
        else if (n < tree->number)
        {
                return search(tree->left);
        }
        else if (n > tree->number)
        {
                return search(tree->right);
        }
        else
        {
                return true;
        }
}
```

Program 6.6: search in a binary-search-tree

So, time complexity here:

- Search = $\mathcal{O}(\log n)$
- Insert = $\mathcal{O}(\log n)$ - need to balance the tree

6.5 Hash Table

Hoping for the best

- Search → $\mathcal{O}(1)$, can actually be $\mathcal{O}(n)$ if we get really unlucky.

6.6 Trie

A tree who nodes are arrays! Time complexity:

- Search = $\mathcal{O}(1)$
- Insert = $\mathcal{O}(1)$

6.7 Queue

First In First Out

- enqueue
- dequeue

6.8 Stack

Last In First Out

- push
- pop

6.9 Dictionary

An abstraction on top of hash table. Has *keys* and *values*.

Chapter 7

Python

7.1 Introduction

```python
# A program that says hello to the world

print("hello, world")
```

Program 7.1: Hello Python

To run: $ `python hello.py`

```python
# get_string and print, with concatenation

from cs50 import get_string

s = get_string("What's your name?\n")
print("hello, " + s)
```

Program 7.2: strings in python

```python
# get_string and print, with multiple arguments

from cs50 import get_string

s = get_string("What's your name?\n")
print("hello,", s)
```

Program 7.3: print function in python

```python
# get_string and print, with format strings

from cs50 import get_string

s = get_string("What's your name?\n")
print(f"hello, {s}")
```

Program 7.4: format strings

```python
# get_int and print

from cs50 import get_int

age = get_int("What's your age?\n")
print(f"You are at least {age * 365} days old.")
```

Program 7.5: integers in python

```python
# Conditions and relational operators

from cs50 import get_int

# Prompt user for x
x = get_int("x: ")

# Prompt user for y
y = get_int("y: ")

# Compare x and y
if x < y:
    print("x is less than y")
elif x > y:
    print("x is greater than y")
else:
    print("x is equal to y")
```

Program 7.6: comparisions in python

```python
# Logical operators

from cs50 import get_string

# Prompt user to agree
s = get_string("Do you agree?\n")

# Check whether agreed
if s == "Y" or s == "y":
    print("Agreed.")
elif s == "N" or s == "n":
    print("Not agreed.")
```

Program 7.7: logical operators in python

```python
# Logical operators, using lists

from cs50 import get_string

# Prompt user to agree
s = get_string("Do you agree?\n")

# Check whether agreed
if s.lower() in ["y", "yes"]:
    print("Agreed.")
elif s.lower() in ["n", "no"]:
    print("Not agreed.")
```

Program 7.8: convert string to lowercase in python

```python
# Loops

while True:
        print("hello, world")
```

Program 7.9: while loop in python

```python
# Better design

for i in range(3):
    print("cough")
```

Program 7.10: for loop and `range` in python

```python
# Abstraction


def main():
    for i in range(3):
        cough()


# Cough once
def cough():
    print("cough")


main()
```

Program 7.11: functions in python

```python
# Abstraction with parameterization


def main():
    cough(3)


# Cough some number of times
def cough(n):
    for i in range(n):
        print("cough")


main()
```

Program 7.12: arguments to functions in python

```python
# Abstraction and scope

from cs50 import get_int


def main():
    i = get_positive_int()
    print(i)


# Prompt user for positive integer
def get_positive_int():
    while True:
        n = get_int("Positive Integer: ")
        if n > 0:
            break
    return n


main()
```

Program 7.13: scopes in python

```python
# Prints a row of 4 question marks with a loop

for i in range(4):
    print("?", end="")
print()
```

Program 7.14: named arguments in python

```python
# Prints a row of 4 question marks without a loop

print("?" * 4)
```

Program 7.15: multiplying a string: pythonic

```python
# Prints a 3-by-3 grid of bricks with loops

for i in range(3):
    for j in range(3):
        print("#", end="")
    print()
```

Program 7.16: nested loops in python

```python
# input and print, with format strings

s = input("What's your name?\n")
print(f"hello, {s}")
```

Program 7.17: input strings in python

```python
# input, int, and print

age = int(input("What's your age?\n"))
print(f"You are at least {age * 365} days old.")
```

Program 7.18: input integers in python

```python
# Integer non-overflow

from time import sleep

# Iteratively double i
i = 1
while True:
    print(i)
    sleep(1)
    i *= 2
```

Program 7.19: overflow in python?

Remark. No limit of ints in python!

```python
# Averages three numbers using a list with append

# Scores
scores = []
scores.append(72)
scores.append(73)
scores.append(33)

# Print average
print(f"Average: {sum(scores) / len(scores)}")
```

Program 7.20: lists in python

```python
# Averages three numbers using a list

# Scores
scores = [72, 73, 33]

# Print average
print(f"Average: {sum(scores) / len(scores)}")
```

Program 7.21: directly using lists in python

```python
# Prints string character by character, indexing into string

from cs50 import get_string

s = get_string("Input:  ")
print("Output: ", end="")
for i in range(len(s)):
    print(s[i], end="")
print()
```

Program 7.22: access characters of a string in python

```python
# Prints string character by character

from cs50 import get_string

s = get_string("Input:  ")
print("Output: ", end="")
for c in s:
    print(c, end="")
print()
```

Program 7.23: accessing characters of a string directly in python

```python
# Uppercases string

from cs50 import get_string

s = get_string("Before: ")
print("After:  ", end="")
print(s.upper())
```

Program 7.24: changing to uppercase in python

```python
# Printing command-line arguments, indexing into argv

from sys import argv

for i in range(len(argv)):
    print(argv[i])
```

Program 7.25: command line arguments in python

```python
# Printing command-line arguments

from sys import argv

for arg in argv:
    print(arg)
```

Program 7.26: directly accessing command line arguments in python

```python
# Exits with explicit value, importing argv and exit

from sys import argv, exit

if len(argv) != 2:
    print("missing command-line argument")
    exit(1)
print(f"hello, {argv[1]}")
exit(0)
```

Program 7.27: exiting on error in python

```python
# Implements linear search for names

import sys

# A list of names
names = ["EMMA", "RODRIGO", "BRIAN", "DAVID"]

# Search for EMMA
if "EMMA" in names:
    print("Found")
    sys.exit(0)
print("Not found")
sys.exit(1)
```

Program 7.28: searching in a list in python

```python
# Implements a phone book

import sys

people = {
    "EMMA": "617-555-0100",
    "RODRIGO": "617-555-0101",
    "BRIAN": "617-555-0102",
    "DAVID": "617-555-0103"
}

# Search for EMMA
if "EMMA" in people:
    print(f"Found {people['EMMA']}")
    sys.exit(0)
print("Not found")
sys.exit(1)
```

Program 7.29: dictionary in python

Remark. A *dictionary* (key/value pair) are also known as associative arrays.

```python
# Compares two strings

from cs50 import get_string

# Get two strings
s = get_string("s: ")
t = get_string("t: ")

# Compare strings
if s == t:
    print("Same")
else:
    print("Different")
```

Program 7.30: string comparision in python

```python
# Swaps two integers

x = 1
y = 2

print(f"x is {x}, y is {y}")
x, y = y, x
print(f"x is {x}, y is {y}")
```

Program 7.31: swapping values in python

```python
# Saves names and numbers to a CSV file

import csv
from cs50 import get_string

# Open CSV file
file = open("phonebook.csv", "a")

# Get name and number
name = get_string("Name: ")
number = get_string("Number: ")

# Print to file
writer = csv.writer(file)
writer.writerow((name, number))

# Close file
file.close()
```

Program 7.32: files in python

```python
# Saves names and numbers to a CSV file

import csv
from cs50 import get_string

# Get name and number
```

```python
name = get_string("Name: ")
number = get_string("Number: ")

# Open CSV file
with open("phonebook.csv", "a") as file:

    # Print to file
    writer = csv.writer(file)
    writer.writerow((name, number))
```

Program 7.33: `with` in python

7.2 Datatypes

- bool
- float
- int
- str ≡ string
- range ≡ sequence of numbers
- list ≡ sequence of mutable values
- tuple ≡ sequence of immutable values
- dict ≡ collection of key/value pairs
- set ≡ collection of unique values
- …

7.3 Previous assignments from C to python

```python
# Blurs an image

from PIL import Image, ImageFilter

# Blur image
before = Image.open("bridge.bmp")
after = before.filter(ImageFilter.BLUR)
after.save("out.bmp")
```

Program 7.34: blur.py: blur an image

```python
# Words in dictionary
words = set()


def check(word):
    """Return true if word is in dictionary else false"""
    if word.lower() in words:
        return True
    else:
        return False


def load(dictionary):
    """Load dictionary into memory, returning true if
    ↪  successful else false"""
    file = open(dictionary, "r")
    for line in file:
        words.add(line.rstrip("\n"))
    file.close()
    return True


def size():
    """Returns number of words in dictionary if loaded else 0
    ↪  if not yet loaded"""
    return len(words)


```

```python
def unload():
    """Unloads dictionary from memory, returning true if
    ↪ successful else false"""
    return True
```

Program 7.35: dictionary.py: implement a dictionary

7.4 Regular Expressions

.	any character
.*	0 or more characters
.+	1 or more characters
?	optional
^	start of input
$	end of input

...

```python
# Logical operators, using regular expressions

import re
from cs50 import get_string

# Prompt user to agree
s = get_string("Do you agree?\n")

# Check whether agreed
if re.search("^y(es)?$", s, re.IGNORECASE):
    print("Agreed.")
elif re.search("^no?$", s, re.IGNORECASE):
    print("Not agreed.")
```

Program 7.36: regex in python

7.5 Fancier stuff: Hardware usage

```python
# Recognizes a greeting

# Get input
words = input("Say something!\n").lower()

# Respond to speech
if "hello" in words:
    print("Hello to you too!")
elif "how are you" in words:
    print("I am well, thanks!")
elif "goodbye" in words:
    print("Goodbye to you too!")
else:
    print("Huh?")
```

Program 7.37: extremely simple AI

```python
# Recognizes a voice
# https://pypi.org/project/SpeechRecognition/

import speech_recognition

# Obtain audio from the microphone
recognizer = speech_recognition.Recognizer()
with speech_recognition.Microphone() as source:
    print("Say something!")
    audio = recognizer.listen(source)

# Recognize speech using Google Speech Recognition
print("Google Speech Recognition thinks you said:")
print(recognizer.recognize_google(audio))
```

Program 7.38: speach recognition in python

```python
# Responds to a greeting
# https://pypi.org/project/SpeechRecognition/

import speech_recognition

# Obtain audio from the microphone
recognizer = speech_recognition.Recognizer()
with speech_recognition.Microphone() as source:
    print("Say something!")
    audio = recognizer.listen(source)

# Recognize speech using Google Speech Recognition
words = recognizer.recognize_google(audio)

# Respond to speech
if "hello" in words:
    print("Hello to you too!")
elif "how are you" in words:
    print("I am well, thanks!")
elif "goodbye" in words:
    print("Goodbye to you too!")
else:
    print("Huh?")
```

Program 7.39: reply with speach recognition in python

```python
# Responds to a name
# https://pypi.org/project/SpeechRecognition/

import re
import speech_recognition

# Obtain audio from the microphone
recognizer = speech_recognition.Recognizer()
with speech_recognition.Microphone() as source:
    print("Say something!")
    audio = recognizer.listen(source)

# Recognize speech using Google Speech Recognition
```

```python
words = recognizer.recognize_google(audio)

# Respond to speech
matches = re.search("my name is (.*)", words)
if matches:
    print(f"Hey, {matches[1]}.")
else:
    print("Hey, you.")
```

Program 7.40: iteractive speach recognition in python

We can:

- Detect all the faces in a photo.
- Recognize a face.
- Create a QR code.

Chapter 8

Database

8.1 csv files

```python
import csv

# Open CSV file
with open("CS50 2019 - Lecture 7 - Favorite TV Shows
↪  (Responses) - Form Responses 1.csv", "r") as file:

    # Create DictReader
    reader = csv.DictReader(file)

    # Iterate over CSV file, printing each title
    for row in reader:
        print(row["title"])
```

Program 8.1: Read a csv file in python

```python
import csv

# For counting favorites
counts = {}

# Open CSV file
with open("CS50 2019 - Lecture 7 - Favorite TV Shows
 ↪  (Responses) - Form Responses 1.csv", "r") as file:

    # Create DictReader
    reader = csv.DictReader(file)

    # Iterate over CSV file
    for row in reader:

        # Force title to lowercase
        title = row["title"].lower()

        # Add title to counts
        if title in counts:
            counts[title] += 1
        else:
            counts[title] = 1

# Print counts
for title, count in counts.items():
    print(title, count, sep=" | ")
```

Program 8.2: Use a dictionary to count in python

```python
import csv

# For counting favorites
counts = {}

# Open CSV file
with open("CS50 2019 - Lecture 7 - Favorite TV Shows
 ↪  (Responses) - Form Responses 1.csv", "r") as file:

    # Create DictReader
    reader = csv.DictReader(file)

    # Iterate over CSV file
    for row in reader:

        # Force title to lowercase
        title = row["title"].lower()

        # Add title to counts
        if title in counts:
            counts[title] += 1
        else:
            counts[title] = 1

# Print counts, sorted by title
for title, count in sorted(counts.items()):
    print(title, count, sep=" | ")
```

Program 8.3: Print sorted dictionary by 'keys' in python

```python
import csv

# For counting favorites
counts = {}

# Open CSV file
with open("CS50 2019 - Lecture 7 - Favorite TV Shows
↪   (Responses) - Form Responses 1.csv", "r") as file:

    # Create DictReader
    reader = csv.DictReader(file)

    # Iterate over CSV file
    for row in reader:

        # Force title to lowercase
        title = row["title"].lower()

        # Add title to counts
        if title in counts:
            counts[title] += 1
        else:
            counts[title] = 1

# Function for comparing items by value
def f(item):
    return item[1]

# Print counts, sorted by key
for title, count in sorted(counts.items(), key=f,
 ↪  reverse=True):
    print(title, count, sep=" | ")
```

Program 8.4: Print sorted dictionary by 'values' in python

```
import csv

# For counting favorites
counts = {}

# Open CSV file
with open("CS50 2019 - Lecture 7 - Favorite TV Shows
 ↪  (Responses) - Form Responses 1.csv", "r") as file:

    # Create DictReader
    reader = csv.DictReader(file)

    # Iterate over CSV file
    for row in reader:

        # Force title to lowercase
        title = row["title"].lower()

        # Add title to counts
        if title in counts:
            counts[title] += 1
        else:
            counts[title] = 1

# Print counts, sorted by key
```

```
for title, count in sorted(counts.items(), key=lambda item:
    item[1], reverse=True):
    print(title, count, sep=" | ")
```

Program 8.5: lambda function in python

8.2 SQL

8.2.1 Example

Open as `sqlite3 <dbname>`:

```
.mode csv
.import <filename> <tablename>
```

Program 8.6: load a csv to a db in sqlite3

Now we can ask the same kind of questions:

```
SELECT title FROM favorites;
SELECT title FROM favorites ORDER BY title;
SELECT title, COUNT(title) FROM favorites GROUP BY title;
SELECT title, COUNT(title) FROM favorites GROUP BY title LIMIT
    10;
SELECT title, COUNT(title) AS n FROM favorites GROUP BY title
    LIMIT 10;
SELECT title, COUNT(title) AS n FROM favorites GROUP BY title
    ORDER BY n DESC LIMIT 10;
```

Program 8.7: SQL querries in sqlite3

8.2.2 Relational Database

With any form of data, there are four fundamental operations:

C: Create

R: Read

U: Update

D: Delete

Structured Query Language is just another programming language mainly used for databases, has keywords attached to these:

1. INSERT
2. SELECT
3. UPDATE
4. DELETE

   ...

8.2.3 Syntax

Datatypes:

1. BLOB - Binary Large Object
2. INTEGER
   (a) smallint
   (b) integer
   (c) bigint
3. NUMERIC
   (a) boolean
   (b) date
   (c) datetime
   (d) numeric(scale,precision)
   (e) time
   (f) timestamp
4. REAL
   (a) real
   (b) double precision
5. TEXT
   (a) char(n)
   (b) varchar(n)
   (c) text

Functions

1. AVG
2. COUNT
3. DISTINCT
4. MAX
5. MIN

   ...

Features

1. WHERE
2. LIKE
3. LIMIT
4. GROUP BY
5. ORDER BY
6. JOIN

   ...

```sql
CREATE TABLE table (column type, ...);
INSERT INTO table (column, ...) VALUES (value, ...);
SELECT columns FROM table;
SELECT title FROM favorites WHERE title LIKE "%office%";
SELECT COUNT(title) FROM favorites WHERE title LIKE "%office%";
SELECT columns FROM table WHERE condition;
UPDATE table SET column=value WHERE condition;
DELETE FROM table WHERE condition;
```

Program 8.8: SQL Syntax

8.2.4 Huge Database

Design decisions really gonna matter. Download "title.basic.tsv.gz" for example.

Fields

1. tcost : tt4786824
2. tytleType : tvSeries
3. primaryTitle : The Crown
4. startYear : 2016
5. genres : Drama, History

```python
import csv

# Open TSV file
# https://datasets.imdbws.com/title.basics.tsv.gz
with open("title.basics.tsv", "r") as titles:

    # Create DictReader
    reader = csv.DictReader(titles, delimiter="\t")

    # Open CSV file
    with open("shows2.csv", "w") as shows:

        # Create writer
        writer = csv.writer(shows)

        # Write header
        writer.writerow(["tconst", "primaryTitle", "startYear",
         ↪  "genres"])

        # Iterate over TSV file
        for row in reader:

            # If non-adult TV show
            if row["titleType"] == "tvSeries" and
             ↪  row["isAdult"] == "0":

                # If year not missing
                if row["startYear"] != "\\N":

                    # Remove \N from genres
                    genres = row["genres"] if row["genres"] != "\\N" else None

                    # If since 1970
                    if int(row["startYear"]) >= 1970:

                        # Write row
                        writer.writerow([row["tconst"], row["primaryTitle"], row["startYear"], genres])
```

Program 8.9: filtering the database in python

```python
import csv

# Prompt user for title
title = input("Title: ")

# Open CSV file
with open("shows2.csv", "r") as input:

    # Create DictReader
    reader = csv.DictReader(input)

    # Iterate over CSV file
    for row in reader:

        # Search for title
        if title.lower() == row["primaryTitle"].lower():
            print(row["primaryTitle"], row["startYear"], row["genres"], sep=" | ")
```

Program 8.10: searching the database in python

```python
import cs50
import csv

# Create database
open("shows3.db", "w").close()
db = cs50.SQL("sqlite:///shows3.db")

# Create table
db.execute("CREATE TABLE shows (tconst TEXT, primaryTitle TEXT,
  ↪  startYear NUMERIC, genres TEXT)")

# Open TSV file
# https://datasets.imdbws.com/title.basics.tsv.gz
with open("title.basics.tsv", "r") as titles:

    # Create DictReader
    reader = csv.DictReader(titles, delimiter="\t")

    # Iterate over TSV file
    for row in reader:

        # If non-adult TV show
        if row["titleType"] == "tvSeries" and row["isAdult"] ==
          ↪  "0":

            # If year not missing
            if row["startYear"] != "\\N":

                # If since 1970
                startYear = int(row["startYear"])
                if startYear >= 1970:

                    # Remove \N from genres
                    genres = row["genres"] if row["genres"] !=
                      ↪  "\\N" else None

                    # Insert show
                    db.execute("INSERT INTO shows (tconst, primaryTitle, startYear, genres) VALUES(?, ?, ?, ?)",
                               row["tconst"], row["primaryTitle"], startYear, genres)
```

Program 8.11: using SQL in python

```python
import cs50
import csv

# Create database
open("shows4.db", "w").close()
db = cs50.SQL("sqlite:///shows4.db")

# Create tables
db.execute("CREATE TABLE shows (id INT, title TEXT, year NUMERIC, PRIMARY KEY(id))")
db.execute("CREATE TABLE genres (show_id INT, genre TEXT, FOREIGN KEY(show_id) REFERENCES shows(id))")

# Open TSV file
# https://datasets.imdbws.com/title.basics.tsv.gz
with open("title.basics.tsv", "r") as titles:

    # Create DictReader
    reader = csv.DictReader(titles, delimiter="\t")

    # Iterate over TSV file
    for row in reader:

        # If non-adult TV show
        if row["titleType"] == "tvSeries" and row["isAdult"] == "0":

            # If year not missing
            if row["startYear"] != "\\N":

                # If since 1970
                startYear = int(row["startYear"])
                if startYear >= 1970:

                    # Trim prefix from tconst
                    id = int(row["tconst"][2:])

                    # Insert show
                    db.execute("INSERT INTO shows (id, title, year) VALUES(?, ?, ?)", id, row["primaryTitle"], startYear)

                    # Insert genres
                    if row["genres"] != "\\N":
                        for genre in row["genres"].split(","):
                            db.execute("INSERT INTO genres (show_id, genre) VALUES(?, ?)", id, genre)
```

Program 8.12: import to multiple tables in SQL using python

```sql
SELECT * FROM shows WHERE id IN (SELECT show_id FROM genres WHERE genre = "Comedy") AND year = 2019;
```

Program 8.13: query with multiple tables in SQL

```sql
CREATE INDEX person_index ON stars (person_id);
```

Program 8.14: indexing in sql

8.3 Problems

8.3.1 Race Conditions

Solution? *Transactions*

8.3.2 SQL Injection Attacks

Solution? *Sanitize your inputs*