

Notes

Introduction to Computer Science (CS50) on EdX

Sparsh Jain

November 26, 2020

Contents

| | | |
|----------|--|----------|
| 1 | Computational Thinking, Scratch | 8 |
| 1.1 | Binary Number System | 8 |
| 1.2 | Algorithms | 8 |
| 1.3 | Time Complexity | 8 |
| 1.4 | Pseudocode | 8 |
| 1.5 | Scratch | 8 |
| 2 | C | 9 |
| 2.1 | Hello World | 9 |
| 2.2 | Input | 9 |
| 2.3 | Initialization | 11 |
| 2.4 | Increment | 11 |
| 2.5 | Conditionals | 11 |
| 2.6 | Loops | 11 |
| 2.6.1 | While Loop | 11 |
| 2.6.2 | For Loop | 12 |
| 2.7 | Additional Info | 12 |
| 2.7.1 | Datatypes | 12 |
| 2.7.2 | Functions | 12 |
| 2.7.3 | Placeholders | 13 |
| 2.7.4 | Arithmetic Operations | 13 |
| 2.8 | Examples | 13 |
| 2.8.1 | Arithmetic | 13 |
| 2.8.2 | Conditional | 16 |
| 2.8.3 | Logical | 17 |
| 2.8.4 | Loop | 18 |
| 2.8.5 | Function | 19 |
| 2.9 | Limitations | 24 |

| | | |
|----------|---|-----------|
| 3 | Arrays | 26 |
| 3.1 | Compiling | 26 |
| 3.1.1 | Preprocessing | 26 |
| 3.1.2 | Compiling | 26 |
| 3.1.3 | Assembling | 26 |
| 3.1.4 | Linking | 26 |
| 3.2 | Debugging | 26 |
| 3.3 | Casting | 27 |
| 3.4 | Array | 27 |
| 3.5 | String | 28 |
| 3.6 | Command Line Arguments | 36 |
| 4 | Algorithms | 38 |
| 4.1 | Linear Search | 38 |
| 4.2 | Binary Search | 38 |
| 4.3 | Efficiency | 39 |
| 4.3.1 | \mathcal{O} Notation: | 39 |
| 4.3.2 | Ω Notation: | 39 |
| 4.4 | Examples | 40 |
| 4.4.1 | Linear Search | 40 |
| 4.4.2 | Bad Design | 41 |
| 4.4.3 | Good Design - <code>typedef struct</code> | 42 |
| 4.5 | Bubble Sort | 43 |
| 4.6 | Selection Sort | 43 |
| 4.7 | Better Bubble Sort | 44 |
| 4.8 | Recursion | 44 |
| 4.9 | Merge Sort | 47 |
| 4.9.1 | Θ Notation | 47 |
| 5 | Memory | 48 |
| 5.1 | Hexadecimal | 48 |
| 5.2 | Addresses | 48 |
| 5.2.1 | Operators | 49 |
| 5.3 | Pointers | 50 |
| 5.4 | Strings | 51 |
| 5.5 | String Comparision | 53 |
| 5.6 | String Copy | 55 |
| 5.7 | Malloc and Free | 56 |
| 5.8 | Buffer Overflow | 56 |
| 5.9 | Swap | 57 |
| 5.10 | scanf | 59 |

| | |
|---|-----------|
| 5.11 File I/O | 60 |
| 6 Data Structures | 63 |
| 6.1 Arrays | 63 |
| 6.2 Data Structures | 66 |
| 6.3 Linked List | 66 |
| 6.4 Tree | 68 |
| 6.4.1 Binary Search Tree | 68 |
| 6.5 Hash Table | 69 |
| 6.6 Trie | 69 |
| 6.7 Queue | 70 |
| 6.8 Stack | 70 |
| 6.9 Dictionary | 70 |
| 7 Python | 71 |
| 7.1 Introduction | 71 |
| 7.2 Datatypes | 82 |
| 7.3 Previous assignments from C to python | 83 |
| 7.4 Regular Expressions | 84 |
| 7.5 Fancier stuff: Hardware usage | 85 |
| 8 Database | 88 |
| 8.1 csv files | 88 |
| 8.2 SQL | 92 |
| 8.2.1 Example | 92 |
| 8.2.2 Relational Database | 92 |
| 8.2.3 Syntax | 93 |
| 8.2.4 Huge Database | 95 |
| 8.3 Problems | 99 |
| 8.3.1 Race Conditions | 99 |
| 8.3.2 SQL Injection Attacks | 99 |

List of Programs

| | | |
|------|----------------------------|----|
| 2.1 | Hello World in C | 9 |
| 2.2 | Hello User in C | 10 |
| 2.3 | int.c | 14 |
| 2.4 | float.c | 14 |
| 2.5 | parity.c | 15 |
| 2.6 | conditions.c | 16 |
| 2.7 | agree.c | 17 |
| 2.8 | cough0.c | 18 |
| 2.9 | cough1.c | 18 |
| 2.10 | cough2.c | 19 |
| 2.11 | cough3.c | 20 |
| 2.12 | positive.c | 21 |
| 2.13 | mario0.c | 21 |
| 2.14 | mario2.c | 22 |
| 2.15 | mario8.c | 23 |
| 2.16 | floats.c | 24 |
| 2.17 | overflow.c | 24 |
| 3.1 | casting | 27 |
| 3.2 | scores0.c | 27 |
| 3.3 | scores1.c | 28 |
| 3.4 | scores2.c | 29 |
| 3.5 | scores3.c | 30 |
| 3.6 | names.c | 31 |
| 3.7 | string0.c | 32 |
| 3.8 | string1.c | 32 |
| 3.9 | string2.c | 33 |
| 3.10 | uppercase0.c | 34 |
| 3.11 | uppercase1.c | 35 |
| 3.12 | argv.c | 36 |
| 3.13 | argv2.c | 37 |
| 3.14 | exit.c | 37 |

| | | |
|------|--|----|
| 4.1 | Linear Search Pseudocode | 38 |
| 4.2 | Binary Search Pseudocode | 38 |
| 4.3 | Linear Search on numbers | 40 |
| 4.4 | Linear Search on names | 41 |
| 4.5 | Linear Search in a phonebook | 42 |
| 4.6 | Linear Search in phonebook with <code>typedef struct</code> | 43 |
| 4.7 | Iteration Pseudocode | 44 |
| 4.8 | Recursion Pseudocode | 45 |
| 4.9 | Iteration C code | 45 |
| 4.10 | Recursion C code | 46 |
| 4.11 | Merge Sort Pseudocode | 47 |
| 5.1 | integer | 48 |
| 5.2 | address of an integer | 49 |
| 5.3 | address2.c | 49 |
| 5.4 | accessing an address | 50 |
| 5.5 | pointers | 50 |
| 5.6 | strings | 51 |
| 5.7 | strings are pointers | 51 |
| 5.8 | strings are <code>char []</code> addresses are consecutive in arrays | 52 |
| 5.9 | accessing characters in a string | 52 |
| 5.10 | accessing characters in a <code>char *</code> | 52 |
| 5.11 | comparing integers | 53 |
| 5.12 | attempting to compare strings directly | 54 |
| 5.13 | comparing strings properly | 54 |
| 5.14 | attempting to copying strings directly | 55 |
| 5.15 | copy strings properly | 56 |
| 5.16 | buffer overflow | 57 |
| 5.17 | naive attempt at swap | 57 |
| 5.18 | swap | 58 |
| 5.19 | scanning an integer | 59 |
| 5.20 | scanning a string in uninitialized | 59 |
| 5.21 | scanning a long string in small array | 60 |
| 5.22 | files in c | 61 |
| 5.23 | phonebook.csv | 61 |
| 5.24 | check jpeg or not | 62 |
| 6.1 | array with hardcoded size | 64 |
| 6.2 | array with dynamic size using malloc | 65 |
| 6.3 | array with dynamic size using realloc | 66 |
| 6.4 | linked list | 68 |

| | | |
|------|---|----|
| 6.5 | node for a binary tree | 68 |
| 6.6 | search in a binary-search-tree | 69 |
| 7.1 | Hello Python | 71 |
| 7.2 | strings in python | 71 |
| 7.3 | print function in python | 71 |
| 7.4 | format strings | 72 |
| 7.5 | integers in python | 72 |
| 7.6 | comparisions in python | 72 |
| 7.7 | logical operators in python | 73 |
| 7.8 | convert string to lowercase in python | 73 |
| 7.9 | while loop in python | 73 |
| 7.10 | for loop and <code>range</code> in python | 74 |
| 7.11 | functions in python | 74 |
| 7.12 | arguments to functions in python | 74 |
| 7.13 | scopes in python | 75 |
| 7.14 | named arguments in python | 75 |
| 7.15 | multiplying a string: pythonic | 75 |
| 7.16 | nested loops in python | 76 |
| 7.17 | input strings in python | 76 |
| 7.18 | input integers in python | 76 |
| 7.19 | overflow in python? | 76 |
| 7.20 | lists in python | 77 |
| 7.21 | directly using lists in python | 77 |
| 7.22 | access characters of a string in python | 77 |
| 7.23 | accessing characters of a string directly in python | 78 |
| 7.24 | changing to uppercase in python | 78 |
| 7.25 | command line arguments in python | 78 |
| 7.26 | directly accessing command line arguments in python | 79 |
| 7.27 | exiting on error in python | 79 |
| 7.28 | searching in a list in python | 79 |
| 7.29 | dictionary in python | 80 |
| 7.30 | string comparision in python | 80 |
| 7.31 | swapping values in python | 81 |
| 7.32 | files in python | 81 |
| 7.33 | <code>with</code> in python | 82 |
| 7.34 | blur.py: blur an image | 83 |
| 7.35 | dictionary.py: implement a dictionary | 84 |
| 7.36 | regex in python | 84 |
| 7.37 | extremely simple AI | 85 |
| 7.38 | speech recognition in python | 85 |

| | | |
|------|---|----|
| 7.39 | reply with speech recognition in python | 86 |
| 7.40 | interactive speech recognition in python | 87 |
| 8.1 | Read a csv file in python | 88 |
| 8.2 | Use a dictionary to count in python | 89 |
| 8.3 | Print sorted dictionary by 'keys' in python | 90 |
| 8.4 | Print sorted dictionary by 'values' in python | 91 |
| 8.5 | lambda function in python | 92 |
| 8.6 | load a csv to a db in sqlite3 | 92 |
| 8.7 | SQL queries in sqlite3 | 92 |
| 8.8 | SQL Syntax | 94 |
| 8.9 | filtering the database in python | 96 |
| 8.10 | searching the database in python | 96 |
| 8.11 | using SQL in python | 98 |
| 8.12 | import to multiple tables in SQL using python | 99 |
| 8.13 | query with multiple tables in SQL | 99 |
| 8.14 | indexing in sql | 99 |

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

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      printf("Hello, World!\n");
6  }
```

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!

```
1  #include <cs50.h>
2  #include <stdio.h>
3
4  int main(void)
5  {
6      string answer = get_string("What's your name?\n");
7      printf("Hello, %s!\n", answer);
8  }
```

Program 2.2: Hello User in C

2.3 Initialization

```
1      int counter = 0;
```

2.4 Increment

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

2.5 Conditionals

```
1      if (x < y)
2      {
3          printf("x is less than y!\n");
4      }
5      else if (x > y)
6      {
7          printf("x is greater than y!\n");
8      }
9      else // if (x == y)
10     {
11         printf("x is equal to y!\n");
12     }
```

2.6 Loops

2.6.1 While Loop

Infinite Loop

```
1      while(true)
2      {
3
4      }
```

Repeat

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

```
3      {
4          printf("Hello World!\n");
5          i = i+1;
6      }
```

2.6.2 For Loop

```
1      for(int i = 0; i < 50; i += 1)
2      {
3          printf("Hello World!\n");
4      }
```

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

```

1  #include <cs50.h>
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int age = get_int("What's your age?\n");
7      // int days = age * 365;
8      // printf("You are atleast %i days old.\n", days);
9      printf("You are atleast %i days old.\n", age * 365);
10 }

```

Program 2.3: int.c

```

1  #include <cs50.h>
2  #include <stdio.h>
3
4  int main(void)
5  {
6      float price = get_float("What's the price?\n");
7      // printf("Your total is %f.\n", price * 1.18);
8      printf("Your total is %.2f.\n", price * 1.18);
9  }

```

Program 2.4: float.c

```

1  #include <cs50.h>
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int n = get_int("n: ");
7
8      if (n % 2 == 0)
9      {
10         printf("even.\n");
11     }
12     else
13     {
14         printf("odd.\n");
15     }
16 }

```

Program 2.5: parity.c

2.8.2 Conditional

```
1 // Conditions and relational operators
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for x
9     int x = get_int("x: ");
10
11     // Prompt user for y
12     int y = get_int("y: ");
13
14     // Compare x and y
15     if (x < y)
16     {
17         printf("x is less than y\n");
18     }
19     else if (x > y)
20     {
21         printf("x is greater than y\n");
22     }
23     else
24     {
25         printf("x is equal to y\n");
26     }
27 }
```

Program 2.6: conditions.c

2.8.3 Logical

```
1  // Logical operators
2  #include <cs50.h>
3  #include <stdio.h>
4  int main(void)
5  {
6      // Prompt user to agree
7      char c = get_char("Do you agree?\n");
8      // Check whether agreed
9      if (c == 'Y'  c == 'y')
10     {
11         printf("Agreed.\n");
12     }
13     else if (c == 'N'  c == 'n')
14     {
15         printf("Not agreed.\n");
16     }
17 }
```

Program 2.7: agree.c

2.8.4 Loop

```
1  // Opportunity for better design
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7      printf("cough\n");
8      printf("cough\n");
9      printf("cough\n");
10 }
```

Program 2.8: cough0.c

```
1  // Better design
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7      for (int i = 0; i < 3; i++)
8      {
9          printf("cough\n");
10     }
11 }
```

Program 2.9: cough1.c

2.8.5 Function

```
1  // Abstraction
2
3  #include <stdio.h>
4
5  void cough(void);
6
7  int main(void)
8  {
9      for (int i = 0; i < 3; i++)
10     {
11         cough();
12     }
13 }
14
15 // Cough once
16 void cough(void)
17 {
18     printf("cough\n");
19 }
```

Program 2.10: cough2.c

```

1  // Abstraction with parameterization
2
3  #include <stdio.h>
4
5  void cough(int n);
6
7  int main(void)
8  {
9      cough(3);
10 }
11
12 // Cough some number of times
13 void cough(int n)
14 {
15     for (int i = 0; i < n; i++)
16     {
17         printf("cough\n");
18     }
19 }

```

Program 2.11: cough3.c

```

1  // Abstraction and scope
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int get_positive_int(void);
7
8  int main(void)
9  {
10     int i = get_positive_int();
11     printf("%i\n", i);
12 }
13
14 // Prompt user for positive integer
15 int get_positive_int(void)
16 {
17     int n;
18     do
19     {
20         n = get_int("Positive Integer: ");
21     }
22     while (n < 1);
23     return n;
24 }

```

Program 2.12: positive.c

```

1  // Prints a row of 4 question marks
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7     printf("????\n");
8 }

```

Program 2.13: mario0.c

```

1  // Prints a row of n question marks with a loop
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      int n;
9      do
10     {
11         n = get_int("Width: ");
12     }
13     while (n < 1);
14     for (int i = 0; i < n; i++)
15     {
16         printf("?");
17     }
18     printf("\n");
19 }

```

Program 2.14: mario2.c

```

1  // Prints an n-by-n grid of bricks with a loop
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      int n;
9      do
10     {
11         n = get_int("Size: ");
12     }
13     while (n < 1);
14     for (int i = 0; i < n; i++)
15     {
16         for (int j = 0; j < n; j++)
17         {
18             printf("#");
19         }
20         printf("\n");
21     }
22 }

```

Program 2.15: mario8.c

2.9 Limitations

```
1  // Floating-point arithmetic with float
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      // Prompt user for x
9      float x = get_float("x: ");
10
11     // Prompt user for y
12     float y = get_float("y: ");
13
14     // Perform division
15     printf("x / y = %.50f\n", x / y);
16 }
```

Program 2.16: floats.c

```
1  // Integer overflow
2
3  #include <stdio.h>
4  #include <unistd.h>
5
6  int main(void)
7  {
8      // Iteratively double i
9      for (int i = 1; ; i *= 2)
10     {
11         printf("%i\n", i);
12         sleep(1);
13     }
14 }
```

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 relevant 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

```
1 // Prints ASCII codes
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     char c1 = 'H';
8     char c2 = 'I';
9     char c3 = '!';
10    printf("%i %i %i\n", c1, c2, c3);
11 }
```

Program 3.1: casting

3.4 Array

Follow through the following examples:

```
1 // Averages three numbers
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Scores
9     int score1 = 72;
10    int score2 = 73;
11    int score3 = 33;
12
13    // Print average
14    printf("Average: %i\n", (score1 + score2 + score3) / 3);
15 }
```

Program 3.2: scores0.c

```

1  // Averages three numbers using an array
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      // Scores
9      int scores[3];
10     scores[0] = 72;
11     scores[1] = 73;
12     scores[2] = 33;
13
14     // Print average
15     printf("Average: %i\n", (scores[0] + scores[1] + scores[2])
16         / 3);
17 }

```

Program 3.3: scores1.c

3.5 String

string is just (or a little more) than an array of chars.

```

1  // Averages three numbers using an array and a constant
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  const int N = 3;
7
8  int main(void)
9  {
10     // Scores
11     int scores[N];
12     scores[0] = 72;
13     scores[1] = 73;
14     scores[2] = 33;
15
16     // Print average
17     printf("Average: %i\n", (scores[0] + scores[1] + scores[2])
18           / N);
19 }

```

Program 3.4: scores2.c

```

1  // Averages numbers using a helper function
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  float average(int length, int array[]);
7
8  int main(void)
9  {
10     // Get number of scores
11     int n = get_int("Scores: ");
12
13     // Get scores
14     int scores[n];
15     for (int i = 0; i < n; i++)
16     {
17         scores[i] = get_int("Score %i: ", i + 1);
18     }
19
20     // Print average
21     printf("Average: %.1f\n", average(n, scores));
22 }
23
24 float average(int length, int array[])
25 {
26     int sum = 0;
27     for (int i = 0; i < length; i++)
28     {
29         sum += array[i];
30     }
31     return (float) sum / (float) length;
32 }

```

Program 3.5: scores3.c

```

1  // Stores names using an array
2
3  #include <cs50.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  int main(void)
8  {
9      // Names
10     string names[4];
11     names[0] = "EMMA";
12     names[1] = "RODRIGO";
13     names[2] = "BRIAN";
14     names[3] = "DAVID";
15
16     // Print Emma's name
17     printf("%s\n", names[0]);
18     printf("%c%c%c%c\n", names[0][0], names[0][1], names[0][2],
19         ↪ names[0][3]);

```

Program 3.6: names.c


```

1 // Prints string char by char, one per line
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     string s = get_string("Input: ");
9     printf("Output: ");
10    for (int i = 0; s[i] != '\0'; i++)
11    {
12        printf("%c", s[i]);
13    }
14    printf("\n");
15 }

```

Program 3.7: string0.c

```

1 // Prints string char by char, one per line, using strlen
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 int main(void)
8 {
9     string s = get_string("Input: ");
10    printf("Output: ");
11    for (int i = 0; i < strlen(s); i++)
12    {
13        printf("%c", s[i]);
14    }
15    printf("\n");
16 }

```

Program 3.8: string1.c

```

1  // Prints string char by char, one per line, using strlen,
   - remembering string's length
2
3  #include <cs50.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  int main(void)
8  {
9      string s = get_string("Input: ");
10     printf("Output: ");
11     for (int i = 0, n = strlen(s); i < n; i++)
12     {
13         printf("%c", s[i]);
14     }
15     printf("\n");
16 }

```

Program 3.9: string2.c

```

1  // Uppercases a string
2
3  #include <cs50.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  int main(void)
8  {
9      string s = get_string("Before: ");
10     printf("After: ");
11     for (int i = 0, n = strlen(s); i < n; i++)
12     {
13         if (s[i] >= 'a' && s[i] <= 'z')
14         {
15             printf("%c", s[i] - 32);
16         }
17         else
18         {
19             printf("%c", s[i]);
20         }
21     }
22     printf("\n");
23 }

```

Program 3.10: uppercase0.c

```

1  // Uppercases string using ctype library (and an unnecessary
   - condition)
2
3  #include <cs50.h>
4  #include <ctype.h>
5  #include <stdio.h>
6  #include <string.h>
7
8  int main(void)
9  {
10     string s = get_string("Before: ");
11     printf("After: ");
12     for (int i = 0, n = strlen(s); i < n; i++)
13     {
14         if (islower(s[i]))
15         {
16             printf("%c", toupper(s[i]));
17         }
18         else
19         {
20             printf("%c", s[i]);
21         }
22     }
23     printf("\n");
24 }

```

Program 3.11: uppercasel.c

3.6 Command Line Arguments

```
1  // Printing a command-line argument
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(int argc, string argv[])
7  {
8      if (argc == 2)
9      {
10         printf("hello, %s\n", argv[1]);
11     }
12     else
13     {
14         printf("hello, world\n");
15     }
16 }
```

Program 3.12: argv.c

```

1  // Printing characters in an array of strings
2
3  #include <cs50.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  int main(int argc, string argv[])
8  {
9      for (int i = 0; i < argc; i++)
10     {
11         for (int j = 0, n = strlen(argv[i]); j < n; j++)
12         {
13             printf("%c\n", argv[i][j]);
14         }
15         printf("\n");
16     }
17 }

```

Program 3.13: argv2.c

```

1  // Returns explicit value from main
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(int argc, string argv[])
7  {
8      if (argc != 2)
9      {
10         printf("missing command-line argument\n");
11         return 1;
12     }
13     printf("hello, %s\n", argv[1]);
14     return 0;
15 }

```

Program 3.14: exit.c

Chapter 4

Algorithms

4.1 Linear Search

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

Program 4.1: Linear Search Pseudocode

4.2 Binary Search

```
1         if no items
2             return false;
3         if middle item is 50
4             return true;
5         else if 50 < middle item
6             search left half
7         else if 50 > middle item
8             search right half
```

Program 4.2: Binary Search Pseudocode

4.3 Efficiency

4.3.1 \mathcal{O} Notation:

Worst case scenario

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

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

```
1 // Implements linear search for numbers
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // An array of numbers
9     int numbers[] = {4, 8, 15, 16, 23, 42};
10
11     // Search for 50
12     for (int i = 0; i < 6; i++)
13     {
14         if (numbers[i] == 50)
15         {
16             printf("Found\n");
17             return 0;
18         }
19     }
20     printf("Not found\n");
21     return 1;
22 }
```

Program 4.3: Linear Search on numbers

Names

```
1 // Implements linear search for names
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 int main(void)
8 {
```

```

9      // An array of names
10     string names[] = {"EMMA", "RODRIGO", "BRIAN", "DAVID"};
11
12     // Search for EMMA
13     for (int i = 0; i < 4; i++)
14     {
15         if (strcmp(names[i], "EMMA") == 0)
16         {
17             printf("Found\n");
18             return 0;
19         }
20     }
21     printf("Not found\n");
22     return 1;
23 }

```

Program 4.4: Linear Search on names

4.4.2 Bad Design

Correct/Working code but bad design!

```

1  // Implements a phone book without structs
2
3  #include <cs50.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  int main(void)
8  {
9      string names[] = {"EMMA", "RODRIGO", "BRIAN", "DAVID"};
10     string numbers[] = {"617-555-0100", "617-555-0101",
11                          "617-555-0102", "617-555-0103"};
12
13     for (int i = 0; i < 4; i++)
14     {
15         if (!strcmp(names[i], "EMMA"))
16         {
17             printf("Found %s\n", numbers[i]);
18             return 0;
19         }
20     }
21 }

```

```

19     }
20     printf("Not found\n");
21     return 1;
22 }

```

Program 4.5: Linear Search in a phonebook

4.4.3 Good Design - typedef struct

Using `typedef struct` for better design!

```

1  // Implements a phone book with structs
2
3  #include <cs50.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  typedef struct
8  {
9      string name;
10     string number;
11 }
12 person;
13
14 int main(void)
15 {
16     person people[4];
17
18     people[0].name = "EMMA";
19     people[0].number = "617-555-0100";
20
21     people[1].name = "RODRIGO";
22     people[1].number = "617-555-0101";
23
24     people[2].name = "BRIAN";
25     people[2].number = "617-555-0102";
26
27     people[3].name = "DAVID";
28     people[3].number = "617-555-0103";
29
30     // Search for EMMA

```

```

31     for (int i = 0; i < 4; i++)
32     {
33         if (strcmp(people[i].name, "EMMA") == 0)
34         {
35             printf("Found %s\n", people[i].number);
36             return 0;
37         }
38     }
39     printf("Not found\n");
40     return 1;
41 }

```

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

4.5 Bubble Sort

```

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

```

$$O(n^2)$$

$$\Omega(n^2)$$

4.6 Selection Sort

```

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

```

$$O(n^2)$$

$$\Omega(n^2)$$

4.7 Better Bubble Sort

```
1 repeat until swap
2     for i = 0 to n-2
3         if ith and i+1th elements out of order
4             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?

```
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     Search left half of book
8 else if Smith is later in book
9     Search right half of book
```

```

10 else
11     Quit

```

Program 4.8: Recursion Pseudocode

```

1  // Draws a pyramid using iteration
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  void draw(int h);
7
8  int main(void)
9  {
10     // Get height of pyramid
11     int height = get_int("Height: ");
12
13     // Draw pyramid
14     draw(height);
15 }
16
17 void draw(int h)
18 {
19     // Draw pyramid of height h
20     for (int i = 1; i <= h; i++)
21     {
22         for (int j = 1; j <= i; j++)
23         {
24             printf("#");
25         }
26         printf("\n");
27     }
28 }

```

Program 4.9: Iteration C code

```

1  // Draws a pyramid using recursion
2
3  #include <cs50.h>
4  #include <stdio.h>

```

```

5
6 void draw(int h);
7
8 int main(void)
9 {
10     // Get height of pyramid
11     int height = get_int("Height: ");
12
13     // Draw pyramid
14     draw(height);
15 }
16
17 void draw(int h)
18 {
19     // If nothing to draw
20     if (h == 0)
21     {
22         return;
23     }
24
25     // Draw pyramid of height h - 1
26     draw(h - 1);
27
28     // Draw one more row of width h
29     for (int i = 0; i < h; i++)
30     {
31         printf("#");
32     }
33     printf("\n");
34 }

```

Program 4.10: Recursion C code

4.9 Merge Sort

```
1  if only 1 item
2      return
3  else
4      sort left half of items
5      sort right half of items
6      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

```
1 // Prints an integer
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     printf("%i\n", n);
9 }
```

Program 5.1: integer

```

1 // Prints an integer's address
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     printf("%p\n", &n);
9 }

```

Program 5.2: address of an integer

```

1 // Prints an integer via its address
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     printf("%i\n", *&n);
9 }

```

Program 5.3: address2.c

5.2.1 Operators

& = Get the address

* = Go to the address

5.3 Pointers

```
1  // Stores and prints an integer's address
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int n = 50;
8      int *p = &n;
9      printf("%p\n", p);
10 }
```

Program 5.4: accessing an address

```
1  // Stores and prints an integer via its address
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int n = 50;
8      int *p = &n;
9      printf("%i\n", *p);
10 }
```

Program 5.5: pointers

5.4 Strings

There are no strings. Strings are just pointers.

```
1 // Prints a string
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     string s = "EMMA";
9     printf("%s\n", s);
10 }
```

Program 5.6: strings

```
1 // Prints a string's address
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     string s = "EMMA";
9     printf("%p\n", s);
10 }
```

Program 5.7: strings are pointers

```
1 // Prints a string's address as well the addresses of its
  ↳ chars
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     string s = "EMMA";
9     printf("%p\n", s);
10    printf("%p\n", &s[0]);
```

```

11     printf("%p\n", &s[1]);
12     printf("%p\n", &s[2]);
13     printf("%p\n", &s[3]);
14     printf("%p\n", &s[4]);
15 }

```

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

```

1  // Prints a string's chars
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      string s = "EMMA";
9      printf("%c\n", s[0]);
10     printf("%c\n", s[1]);
11     printf("%c\n", s[2]);
12     printf("%c\n", s[3]);
13 }

```

Program 5.9: accessing characters in a string

```

1  // Stores and prints a string's address via pointer arithmetic
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7      char *s = "EMMA";
8      printf("%c\n", *s);
9      printf("%c\n", *(s+1));
10     printf("%c\n", *(s+2));
11     printf("%c\n", *(s+3));
12 }

```

Program 5.10: accessing characters in a `char *`

5.5 String Comparision

```
1  // Compares two integers
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      // Get two integers
9      int i = get_int("i: ");
10     int j = get_int("j: ");
11
12     // Compare integers
13     if (i == j)
14     {
15         printf("Same\n");
16     }
17     else
18     {
19         printf("Different\n");
20     }
21 }
```

Program 5.11: comparing integers

```
1  // Compares two strings' addresses
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      // Get two strings
9      string s = get_string("s: ");
10     string t = get_string("t: ");
11
12     // Compare strings' addresses
13     if (s == t)
14     {
15         printf("Same\n");
```

```

16     }
17     else
18     {
19         printf("Different\n");
20     }
21 }

```

Program 5.12: attempting to compare strings directly

```

1  // Compares two strings using strcmp
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      // Get two strings
9      string s = get_string("s: ");
10     string t = get_string("t: ");
11
12     // Compare strings
13     if (strcmp(s, t) == 0)
14     {
15         printf("Same\n");
16     }
17     else
18     {
19         printf("Different\n");
20     }
21 }

```

Program 5.13: comparing strings properly

5.6 String Copy

```
1 // Capitalizes a string
2
3 #include <cs50.h>
4 #include <ctype.h>
5 #include <stdio.h>
6 #include <string.h>
7
8 int main(void)
9 {
10     // Get a string
11     string s = get_string("s: ");
12
13     // Copy string's address
14     string t = s;
15
16     // Capitalize first letter in string
17     if (strlen(t) > 0)
18     {
19         t[0] = toupper(t[0]);
20     }
21
22     // Print string twice
23     printf("s: %s\n", s);
24     printf("t: %s\n", t);
25 }
```

Program 5.14: attempting to copying strings directly

```
1 // Capitalizes a copy of a string
2
3 #include <cs50.h>
4 #include <ctype.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <string.h>
8
9 int main(void)
10 {
11     // Get a string
```



```

12     char *s = get_string("s: ");
13
14     // Allocate memory for another string
15     char *t = malloc(strlen(s) + 1);
16
17     // Copy string into memory
18     for (int i = 0, n = strlen(s); i <= n; i++)
19     {
20         t[i] = s[i];
21     }
22
23     // Capitalize copy
24     t[0] = toupper(t[0]);
25
26     // Print strings
27     printf("s: %s\n", s);
28     printf("t: %s\n", t);
29 }

```

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

```

1 // http://valgrind.org/docs/manual/quick-start.html
  ↳ #quick-start.prepare
2
3 #include <stdlib.h>
4
5 void f(void)
6 {
7     int *x = malloc(10 * sizeof(int));
8     x[10] = 0;

```

```

9   }
10
11  int main(void)
12  {
13      f();
14      return 0;
15  }

```

Program 5.16: buffer overflow

5.9 Swap

Pass by *value* vs pass by *reference*

```

1  // Fails to swap two integers
2
3  #include <stdio.h>
4
5  void swap(int a, int b);
6
7  int main(void)
8  {
9      int x = 1;
10     int y = 2;
11
12     printf("x is %i, y is %i\n", x, y);
13     swap(x, y);
14     printf("x is %i, y is %i\n", x, y);
15 }
16
17 void swap(int a, int b)
18 {
19     int tmp = a;
20     a = b;
21     b = tmp;
22 }

```

Program 5.17: naive attempt at swap

```

1  // Swaps two integers using pointers
2
3  #include <stdio.h>
4
5  void swap(int *a, int *b);
6
7  int main(void)
8  {
9      int x = 1;
10     int y = 2;
11
12     printf("x is %i, y is %i\n", x, y);
13     swap(&x, &y);
14     printf("x is %i, y is %i\n", x, y);
15 }
16
17 void swap(int *a, int *b)
18 {
19     int tmp = *a;
20     *a = *b;
21     *b = tmp;
22 }

```

Program 5.18: swap

5.10 scanf

```
1 // Gets an int from user using scanf
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int x;
8     printf("x: ");
9     scanf("%i", &x);
10    printf("x: %i\n", x);
11 }
```

Program 5.19: scanning an integer

```
1 // Incorrectly gets a string from user using scanf
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     char *s;
8     printf("s: ");
9     scanf("%s", s);
10    printf("s: %s\n", s);
11 }
```

Program 5.20: scanning a string in uninitialized

```

1  // Dangerously gets a string from user using scanf
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7      char s[5];
8      printf("s: ");
9      scanf("%s", s);
10     printf("s: %s\n", s);
11 }

```

Program 5.21: scanning a long string in small array

5.11 File I/O

```

1  // Saves names and numbers to a CSV file
2
3  #include <cs50.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  int main(void)
8  {
9      // Open CSV file
10     FILE *file = fopen("phonebook.csv", "a");
11     if (!file)
12     {
13         return 1;
14     }
15
16     // Get name and number
17     string name = get_string("Name: ");
18     string number = get_string("Number: ");
19
20     // Print to file
21     fprintf(file, "%s,%s\n", name, number);
22

```

```

23     // Close file
24     fclose(file);
25 }

```

Program 5.22: files in c

```

1 Sparsh,6238-098-518

```

Program 5.23: phonebook.csv

```

1  // Detects if a file is a JPEG
2
3  #include <stdio.h>
4
5  int main(int argc, char *argv[])
6  {
7      // Check usage
8      if (argc != 2)
9      {
10         return 1;
11     }
12
13     // Open file
14     FILE *file = fopen(argv[1], "r");
15     if (!file)
16     {
17         return 1;
18     }
19
20     // Read first three bytes
21     unsigned char bytes[3];
22     fread(bytes, 3, 1, file);
23
24     // Check first three bytes
25     if (bytes[0] == 0xff && bytes[1] == 0xd8 && bytes[2] ==
        0xff)
26     {
27         printf("Maybe\n");
28     }
29     else

```

```
30     {
31         printf("No\n");
32     }
33
34     // Close file
35     fclose(file);
36 }
```

Program 5.24: check jpeg or not

Chapter 6

Data Structures

6.1 Arrays

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

```
1 // Implements a list of numbers with an array of fixed size
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     // List of size 3
8     int list[3];
9
10    // Initialize list with numbers
11    list[0] = 1;
12    list[1] = 2;
13    list[2] = 3;
14
15    // Print list
16    for (int i = 0; i < 3; i++)
17    {
18        printf("%i\n", list[i]);
```



```
19     }
20 }
```

Program 6.1: array with hardcoded size

```
1  // Implements a list of numbers with an array of dynamic size
2  //
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main(void)
7  {
8      // List of size 3
9      int *list = malloc(3 * sizeof(int));
10     if (list == NULL)
11     {
12         return 1;
13     }
14
15     // Initialize list of size 3 with numbers
16     list[0] = 1;
17     list[1] = 2;
18     list[2] = 3;
19
20     // List of size 4
21     int *tmp = malloc(4 * sizeof(int));
22     if (tmp == NULL)
23     {
24         return 1;
25     }
26
27     // Copy list of size 3 into list of size 4
28     for (int i = 0; i < 3; i++)
29     {
30         tmp[i] = list[i];
31     }
32
33     // Add number to list of size 4
34     tmp[3] = 4;
35
36     // Free list of size 3
```

```

37     free(list);
38
39     // Remember list of size 4
40     list = tmp;
41
42     // Print list
43     for (int i = 0; i < 4; i++)
44     {
45         printf("%i\n", list[i]);
46     }
47
48     // Free list
49     free(list);
50 }

```

Program 6.2: array with dynamic size using malloc

```

1  // Implements a list of numbers with an array of dynamic size
   ↳ using realloc
2
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  int main(void)
7  {
8      // List of size 3
9      int *list = malloc(3 * sizeof(int));
10     if (list == NULL)
11     {
12         return 1;
13     }
14
15     // Initialize list of size 3 with numbers
16     list[0] = 1;
17     list[1] = 2;
18     list[2] = 3;
19
20     // Resize list to be of size 4
21     int *tmp = realloc(list, 4 * sizeof(int));
22     if (tmp == NULL)
23     {

```

```

24         return 1;
25     }
26     list = tmp;
27
28     // Add number to list
29     list[3] = 4;
30
31     // Print list
32     for (int i = 0; i < 4; i++)
33     {
34         printf("%i\n", list[i]);
35     }
36
37     // Free list
38     free(list);
39 }

```

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

```

1  // Implements a list of numbers with linked list
2
3  #include <stdio.h>
4  #include <stdlib.h>
5
6  // Represents a node
7  typedef struct node
8  {
9      int number;

```

```

10     struct node *next;
11 }
12 node;
13
14 int main(void)
15 {
16     // List of size 0
17     node *list = NULL;
18
19     // Add number to list
20     node *n = malloc(sizeof(node));
21     if (n == NULL)
22     {
23         return 1;
24     }
25     n->number = 1;
26     n->next = NULL;
27     list = n;
28
29     // Add number to list
30     n = malloc(sizeof(node));
31     if (n == NULL)
32     {
33         return 1;
34     }
35     n->number = 2;
36     n->next = NULL;
37     list->next = n;
38
39     // Add number to list
40     n = malloc(sizeof(node));
41     if (n == NULL)
42     {
43         return 1;
44     }
45     n->number = 3;
46     n->next = NULL;
47     list->next->next = n;
48
49     // Print list
50     for (node *tmp = list; tmp != NULL; tmp = tmp->next)

```

```

51     {
52         printf("%i\n", tmp->number);
53     }
54
55     // Free list
56     while (list != NULL)
57     {
58         node *tmp = list->next;
59         free(list);
60         list = tmp;
61     }
62 }

```

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

```

1  typedef struct node
2  {
3      int number;
4      struct node *left;
5      struct node *right;
6  }
7  node;

```

Program 6.5: node for a binary tree

```

1  bool search(node *tree, int n)
2  {
3      if (tree == NULL)
4      {
5          return false;
6      }
7      else if (n < tree->number)
8      {
9          return search(tree->left);
10     }
11     else if (n > tree->number)
12     {
13         return search(tree->right);
14     }
15     else
16     {
17         return true;
18     }
19 }

```

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 $\rightarrow \mathcal{O}(1)$, 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

```
1  # A program that says hello to the world
2
3  print("hello, world")
```

Program 7.1: Hello Python

To run: \$ python hello.py

```
1  # get_string and print, with concatenation
2
3  from cs50 import get_string
4
5  s = get_string("What's your name?\n")
6  print("hello, " + s)
```

Program 7.2: strings in python

```
1  # get_string and print, with multiple arguments
2
3  from cs50 import get_string
4
5  s = get_string("What's your name?\n")
6  print("hello,", s)
```

Program 7.3: print function in python


```

1  # get_string and print, with format strings
2
3  from cs50 import get_string
4
5  s = get_string("What's your name?\n")
6  print(f"hello, {s}")

```

Program 7.4: format strings

```

1  # get_int and print
2
3  from cs50 import get_int
4
5  age = get_int("What's your age?\n")
6  print(f"You are at least {age * 365} days old.")

```

Program 7.5: integers in python

```

1  # Conditions and relational operators
2
3  from cs50 import get_int
4
5  # Prompt user for x
6  x = get_int("x: ")
7
8  # Prompt user for y
9  y = get_int("y: ")
10
11 # Compare x and y
12 if x < y:
13     print("x is less than y")
14 elif x > y:
15     print("x is greater than y")
16 else:
17     print("x is equal to y")

```

Program 7.6: comparisons in python

```

1  # Logical operators
2
3  from cs50 import get_string
4
5  # Prompt user to agree
6  s = get_string("Do you agree?\n")
7
8  # Check whether agreed
9  if s == "Y" or s == "y":
10     print("Agreed.")
11 elif s == "N" or s == "n":
12     print("Not agreed.")

```

Program 7.7: logical operators in python

```

1  # Logical operators, using lists
2
3  from cs50 import get_string
4
5  # Prompt user to agree
6  s = get_string("Do you agree?\n")
7
8  # Check whether agreed
9  if s.lower() in ["y", "yes"]:
10     print("Agreed.")
11 elif s.lower() in ["n", "no"]:
12     print("Not agreed.")

```

Program 7.8: convert string to lowercase in python

```

1  # Loops
2
3  while True:
4     print("hello, world")

```

Program 7.9: while loop in python

```

1  # Better design
2
3  for i in range(3):
4      print("cough")

```

Program 7.10: for loop and `range` in python

```

1  # Abstraction
2
3
4  def main():
5      for i in range(3):
6          cough()
7
8
9  # Cough once
10 def cough():
11     print("cough")
12
13
14 main()

```

Program 7.11: functions in python

```

1  # Abstraction with parameterization
2
3
4  def main():
5      cough(3)
6
7
8  # Cough some number of times
9  def cough(n):
10     for i in range(n):
11         print("cough")
12
13
14 main()

```

Program 7.12: arguments to functions in python

```

1  # Abstraction and scope
2
3  from cs50 import get_int
4
5
6  def main():
7      i = get_positive_int()
8      print(i)
9
10
11 # Prompt user for positive integer
12 def get_positive_int():
13     while True:
14         n = get_int("Positive Integer: ")
15         if n > 0:
16             break
17     return n
18
19
20 main()

```

Program 7.13: scopes in python

```

1  # Prints a row of 4 question marks with a loop
2
3  for i in range(4):
4      print("?", end=" ")
5  print()

```

Program 7.14: named arguments in python

```

1  # Prints a row of 4 question marks without a loop
2
3  print("?" * 4)

```

Program 7.15: multiplying a string: pythonic

```

1  # Prints a 3-by-3 grid of bricks with loops
2
3  for i in range(3):
4      for j in range(3):
5          print("#", end="")
6      print()

```

Program 7.16: nested loops in python

```

1  # input and print, with format strings
2
3  s = input("What's your name?\n")
4  print(f"hello, {s}")

```

Program 7.17: input strings in python

```

1  # input, int, and print
2
3  age = int(input("What's your age?\n"))
4  print(f"You are at least {age * 365} days old.")

```

Program 7.18: input integers in python

```

1  # Integer non-overflow
2
3  from time import sleep
4
5  # Iteratively double i
6  i = 1
7  while True:
8      print(i)
9      sleep(1)
10     i *= 2

```

Program 7.19: overflow in python?

Remark. No limit of ints in python!

```

1  # Averages three numbers using a list with append
2
3  # Scores
4  scores = []
5  scores.append(72)
6  scores.append(73)
7  scores.append(33)
8
9  # Print average
10 print(f"Average: {sum(scores) / len(scores)}")

```

Program 7.20: lists in python

```

1  # Averages three numbers using a list
2
3  # Scores
4  scores = [72, 73, 33]
5
6  # Print average
7  print(f"Average: {sum(scores) / len(scores)}")

```

Program 7.21: directly using lists in python

```

1  # Prints string character by character, indexing into string
2
3  from cs50 import get_string
4
5  s = get_string("Input: ")
6  print("Output: ", end="")
7  for i in range(len(s)):
8      print(s[i], end="")
9  print()

```

Program 7.22: access characters of a string in python

```

1  # Prints string character by character
2
3  from cs50 import get_string
4
5  s = get_string("Input: ")
6  print("Output: ", end="")
7  for c in s:
8      print(c, end="")
9  print()

```

Program 7.23: accessing characters of a string directly in python

```

1  # Uppercases string
2
3  from cs50 import get_string
4
5  s = get_string("Before: ")
6  print("After: ", end="")
7  print(s.upper())

```

Program 7.24: changing to uppercase in python

```

1  # Printing command-line arguments, indexing into argv
2
3  from sys import argv
4
5  for i in range(len(argv)):
6      print(argv[i])

```

Program 7.25: command line arguments in python

```

1  # Printing command-line arguments
2
3  from sys import argv
4
5  for arg in argv:
6      print(arg)

```

Program 7.26: directly accessing command line arguments in python

```

1  # Exits with explicit value, importing argv and exit
2
3  from sys import argv, exit
4
5  if len(argv) != 2:
6      print("missing command-line argument")
7      exit(1)
8  print(f"hello, {argv[1]}")
9  exit(0)

```

Program 7.27: exiting on error in python

```

1  # Implements linear search for names
2
3  import sys
4
5  # A list of names
6  names = ["EMMA", "RODRIGO", "BRIAN", "DAVID"]
7
8  # Search for EMMA
9  if "EMMA" in names:
10     print("Found")
11     sys.exit(0)
12 print("Not found")
13 sys.exit(1)

```

Program 7.28: searching in a list in python


```

1  # Implements a phone book
2
3  import sys
4
5  people = {
6      "EMMA": "617-555-0100",
7      "RODRIGO": "617-555-0101",
8      "BRIAN": "617-555-0102",
9      "DAVID": "617-555-0103"
10 }
11
12 # Search for EMMA
13 if "EMMA" in people:
14     print(f"Found {people['EMMA']}")
15     sys.exit(0)
16 print("Not found")
17 sys.exit(1)

```

Program 7.29: dictionary in python

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

```

1  # Compares two strings
2
3  from cs50 import get_string
4
5  # Get two strings
6  s = get_string("s: ")
7  t = get_string("t: ")
8
9  # Compare strings
10 if s == t:
11     print("Same")
12 else:
13     print("Different")

```

Program 7.30: string comparision in python

```

1  # Swaps two integers
2
3  x = 1
4  y = 2
5
6  print(f"x is {x}, y is {y}")
7  x, y = y, x
8  print(f"x is {x}, y is {y}")

```

Program 7.31: swapping values in python

```

1  # Saves names and numbers to a CSV file
2
3  import csv
4  from cs50 import get_string
5
6  # Open CSV file
7  file = open("phonebook.csv", "a")
8
9  # Get name and number
10 name = get_string("Name: ")
11 number = get_string("Number: ")
12
13 # Print to file
14 writer = csv.writer(file)
15 writer.writerow((name, number))
16
17 # Close file
18 file.close()

```

Program 7.32: files in python

```

1  # Saves names and numbers to a CSV file
2
3  import csv
4  from cs50 import get_string
5
6  # Get name and number

```

```

7  name = get_string("Name: ")
8  number = get_string("Number: ")
9
10 # Open CSV file
11 with open("phonebook.csv", "a") as file:
12
13     # Print to file
14     writer = csv.writer(file)
15     writer.writerow((name, number))

```

Program 7.33: `with` in python

7.2 Datatypes

- `bool`
- `float`
- `int`
- `str` \equiv `string`
- `range` \equiv sequence of numbers
- `list` \equiv sequence of mutable values
- `tuple` \equiv sequence of immutable values
- `dict` \equiv collection of key/value pairs
- `set` \equiv collection of unique values
- ...

7.3 Previous assignments from C to python

```
1  # Blurs an image
2
3  from PIL import Image, ImageFilter
4
5  # Blur image
6  before = Image.open("bridge.bmp")
7  after = before.filter(ImageFilter.BLUR)
8  after.save("out.bmp")
```

Program 7.34: blur.py: blur an image

```
1  # Words in dictionary
2  words = set()
3
4
5  def check(word):
6      """Return true if word is in dictionary else false"""
7      if word.lower() in words:
8          return True
9      else:
10         return False
11
12
13 def load(dictionary):
14     """Load dictionary into memory, returning true if
15     ↪ successful else false"""
16     file = open(dictionary, "r")
17     for line in file:
18         words.add(line.rstrip("\n"))
19     file.close()
20     return True
21
22 def size():
23     """Returns number of words in dictionary if loaded else 0
24     ↪ if not yet loaded"""
25     return len(words)
26
```

```

27 def unload():
28     """Unloads dictionary from memory, returning true if
        ↳ successful else false"""
29     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

...

```

1  # Logical operators, using regular expressions
2
3  import re
4  from cs50 import get_string
5
6  # Prompt user to agree
7  s = get_string("Do you agree?\n")
8
9  # Check whether agreed
10 if re.search("^y(es)?$", s, re.IGNORECASE):
11     print("Agreed.")
12 elif re.search("^no?$", s, re.IGNORECASE):
13     print("Not agreed.")

```

Program 7.36: regex in python

7.5 Fancier stuff: Hardware usage

```
1  # Recognizes a greeting
2
3  # Get input
4  words = input("Say something!\n").lower()
5
6  # Respond to speech
7  if "hello" in words:
8      print("Hello to you too!")
9  elif "how are you" in words:
10     print("I am well, thanks!")
11  elif "goodbye" in words:
12     print("Goodbye to you too!")
13  else:
14     print("Huh?")
```

Program 7.37: extremely simple AI

```
1  # Recognizes a voice
2  # https://pypi.org/project/SpeechRecognition/
3
4  import speech_recognition
5
6  # Obtain audio from the microphone
7  recognizer = speech_recognition.Recognizer()
8  with speech_recognition.Microphone() as source:
9      print("Say something!")
10     audio = recognizer.listen(source)
11
12  # Recognize speech using Google Speech Recognition
13  print("Google Speech Recognition thinks you said:")
14  print(recognizer.recognize_google(audio))
```

Program 7.38: speech recognition in python

```

1  # Responds to a greeting
2  # https://pypi.org/project/SpeechRecognition/
3
4  import speech_recognition
5
6  # Obtain audio from the microphone
7  recognizer = speech_recognition.Recognizer()
8  with speech_recognition.Microphone() as source:
9      print("Say something!")
10     audio = recognizer.listen(source)
11
12 # Recognize speech using Google Speech Recognition
13 words = recognizer.recognize_google(audio)
14
15 # Respond to speech
16 if "hello" in words:
17     print("Hello to you too!")
18 elif "how are you" in words:
19     print("I am well, thanks!")
20 elif "goodbye" in words:
21     print("Goodbye to you too!")
22 else:
23     print("Huh?")

```

Program 7.39: reply with speech recognition in python

```

1  # Responds to a name
2  # https://pypi.org/project/SpeechRecognition/
3
4  import re
5  import speech_recognition
6
7  # Obtain audio from the microphone
8  recognizer = speech_recognition.Recognizer()
9  with speech_recognition.Microphone() as source:
10     print("Say something!")
11     audio = recognizer.listen(source)
12
13 # Recognize speech using Google Speech Recognition

```

```

14 words = recognizer.recognize_google(audio)
15
16 # Respond to speech
17 matches = re.search("my name is (.*)", words)
18 if matches:
19     print(f"Hey, {matches[1]}.")
20 else:
21     print("Hey, you.")

```

Program 7.40: interactive speech 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

```
1 import csv
2
3 # Open CSV file
4 with open("CS50 2019 - Lecture 7 - Favorite TV Shows
  ↳ (Responses) - Form Responses 1.csv", "r") as file:
5
6     # Create DictReader
7     reader = csv.DictReader(file)
8
9     # Iterate over CSV file, printing each title
10    for row in reader:
11        print(row["title"])
```

Program 8.1: Read a csv file in python

```

1  import csv
2
3  # For counting favorites
4  counts = {}
5
6  # Open CSV file
7  with open("CS50 2019 - Lecture 7 - Favorite TV Shows
    - (Responses) - Form Responses 1.csv", "r") as file:
8
9      # Create DictReader
10     reader = csv.DictReader(file)
11
12     # Iterate over CSV file
13     for row in reader:
14
15         # Force title to lowercase
16         title = row["title"].lower()
17
18         # Add title to counts
19         if title in counts:
20             counts[title] += 1
21         else:
22             counts[title] = 1
23
24     # Print counts
25     for title, count in counts.items():
26         print(title, count, sep=" | ")

```

Program 8.2: Use a dictionary to count in python

```

1  import csv
2
3  # For counting favorites
4  counts = {}
5
6  # Open CSV file
7  with open("CS50 2019 - Lecture 7 - Favorite TV Shows
    - (Responses) - Form Responses 1.csv", "r") as file:
8

```

```

9      # Create DictReader
10     reader = csv.DictReader(file)
11
12     # Iterate over CSV file
13     for row in reader:
14
15         # Force title to lowercase
16         title = row["title"].lower()
17
18         # Add title to counts
19         if title in counts:
20             counts[title] += 1
21         else:
22             counts[title] = 1
23
24     # Print counts, sorted by title
25     for title, count in sorted(counts.items()):
26         print(title, count, sep=" | ")

```

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

```

1  import csv
2
3  # For counting favorites
4  counts = {}
5
6  # Open CSV file
7  with open("CS50 2019 - Lecture 7 - Favorite TV Shows
   ↳ (Responses) - Form Responses 1.csv", "r") as file:
8
9      # Create DictReader
10     reader = csv.DictReader(file)
11
12     # Iterate over CSV file
13     for row in reader:
14
15         # Force title to lowercase
16         title = row["title"].lower()
17
18         # Add title to counts
19         if title in counts:

```

```

20         counts[title] += 1
21     else:
22         counts[title] = 1
23
24     # Function for comparing items by value
25     def f(item):
26         return item[1]
27
28     # Print counts, sorted by key
29     for title, count in sorted(counts.items(), key=f,
30                               ↪ reverse=True):
31         print(title, count, sep=" | ")

```

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

```

1  import csv
2
3  # For counting favorites
4  counts = {}
5
6  # Open CSV file
7  with open("CS50 2019 - Lecture 7 - Favorite TV Shows
8  ↪ (Responses) - Form Responses 1.csv", "r") as file:
9
10     # Create DictReader
11     reader = csv.DictReader(file)
12
13     # Iterate over CSV file
14     for row in reader:
15
16         # Force title to lowercase
17         title = row["title"].lower()
18
19         # Add title to counts
20         if title in counts:
21             counts[title] += 1
22         else:
23             counts[title] = 1
24
25     # Print counts, sorted by key

```

```

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

```

Program 8.5: lambda function in python

8.2 SQL

8.2.1 Example

Open as sqlite3 <dbname>:

```

1 .mode csv
2 .import <filename> <tablename>

```

Program 8.6: load a csv to a db in sqlite3

Now we can ask the same kind of questions:

```

1 SELECT title FROM favorites;
2 SELECT title FROM favorites ORDER BY title;
3 SELECT title, COUNT(title) FROM favorites GROUP BY title;
4 SELECT title, COUNT(title) FROM favorites GROUP BY title LIMIT
    ↪ 10;
5 SELECT title, COUNT(title) AS n FROM favorites GROUP BY title
    ↪ LIMIT 10;
6 SELECT title, COUNT(title) AS n FROM favorites GROUP BY title
    ↪ ORDER BY n DESC LIMIT 10;

```

Program 8.7: SQL queries 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
- ...

```
1 CREATE TABLE table (column type, ...);
2 INSERT INTO table (column, ...) VALUES (value, ...);
3 SELECT columns FROM table;
4 SELECT title FROM favorites WHERE title LIKE "%office%";
5 SELECT COUNT(title) FROM favorites WHERE title LIKE "%office%";
6 SELECT columns FROM table WHERE condition;
7 UPDATE table SET column=value WHERE condition;
8 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. tyleType : tvSeries
3. primaryTitle : The Crown
4. startYear : 2016
5. genres : Drama, History

```
1  import csv
2
3  # Open TSV file
4  # https://datasets.imdbws.com/title.basics.tsv.gz
5  with open("title.basics.tsv", "r") as titles:
6
7      # Create DictReader
8      reader = csv.DictReader(titles, delimiter="\t")
9
10     # Open CSV file
11     with open("shows2.csv", "w") as shows:
12
13         # Create writer
14         writer = csv.writer(shows)
15
16         # Write header
17         writer.writerow(["tconst", "primaryTitle", "startYear",
18             ↪ "genres"])
19
20         # Iterate over TSV file
21         for row in reader:
22
23             # If non-adult TV show
24             if row["titleType"] == "tvSeries" and
25                 ↪ row["isAdult"] == "0":
```



```

24
25     # If year not missing
26     if row["startYear"] != "\\N":
27
28         # Remove \N from genres
29         genres = row["genres"] if row["genres"] !=
            ↪ "\\N" else None
30
31         # If since 1970
32         if int(row["startYear"]) >= 1970:
33
34             # Write row
35             writer.writerow([row["tconst"],
                ↪ row["primaryTitle"],
                ↪ row["startYear"], genres])

```

Program 8.9: filtering the database in python

```

1  import csv
2
3  # Prompt user for title
4  title = input("Title: ")
5
6  # Open CSV file
7  with open("shows2.csv", "r") as input:
8
9      # Create DictReader
10     reader = csv.DictReader(input)
11
12     # Iterate over CSV file
13     for row in reader:
14
15         # Search for title
16         if title.lower() == row["primaryTitle"].lower():
17             print(row["primaryTitle"], row["startYear"],
                ↪ row["genres"], sep=" | ")

```

Program 8.10: searching the database in python

```

1  import cs50
2  import csv
3
4  # Create database
5  open("shows3.db", "w").close()
6  db = cs50.SQL("sqlite:///shows3.db")
7
8  # Create table
9  db.execute("CREATE TABLE shows (tconst TEXT, primaryTitle TEXT,
10             \t startYear NUMERIC, genres TEXT)")
11
12 # Open TSV file
13 # https://datasets.imdbws.com/title.basics.tsv.gz
14 with open("title.basics.tsv", "r") as titles:
15
16     # Create DictReader
17     reader = csv.DictReader(titles, delimiter="\t")
18
19     # Iterate over TSV file
20     for row in reader:
21
22         # If non-adult TV show
23         if row["titleType"] == "tvSeries" and row["isAdult"] ==
24             \t "0":
25
26             # If year not missing
27             if row["startYear"] != "\\N":
28
29                 # If since 1970
30                 startYear = int(row["startYear"])
31                 if startYear >= 1970:
32
33                     # Remove \N from genres
34                     genres = row["genres"] if row["genres"] !=
35                         \t "\\N" else None
36
37                     # Insert show

```

```

35         db.execute("INSERT INTO shows (tconst,
        ↪ primaryTitle, startYear, genres)
        ↪ VALUES(?, ?, ?, ?)",
36                 row["tconst"],
        ↪ row["primaryTitle"],
        ↪ startYear, genres)

```

Program 8.11: using SQL in python

```

1  import cs50
2  import csv
3
4  # Create database
5  open("shows4.db", "w").close()
6  db = cs50.SQL("sqlite:///shows4.db")
7
8  # Create tables
9  db.execute("CREATE TABLE shows (id INT, title TEXT, year
    ↪ NUMERIC, PRIMARY KEY(id))")
10 db.execute("CREATE TABLE genres (show_id INT, genre TEXT,
    ↪ FOREIGN KEY(show_id) REFERENCES shows(id))")
11
12 # Open TSV file
13 # https://datasets.imdbws.com/title.basics.tsv.gz
14 with open("title.basics.tsv", "r") as titles:
15
16     # Create DictReader
17     reader = csv.DictReader(titles, delimiter="\t")
18
19     # Iterate over TSV file
20     for row in reader:
21
22         # If non-adult TV show
23         if row["titleType"] == "tvSeries" and row["isAdult"] ==
            ↪ "0":
24
25             # If year not missing
26             if row["startYear"] != "\\N":
27
28                 # If since 1970
29                 startYear = int(row["startYear"])

```

```

30         if startYear >= 1970:
31
32             # Trim prefix from tconst
33             id = int(row["tconst"][2:])
34
35             # Insert show
36             db.execute("INSERT INTO shows (id, title,
37                 ↳ year) VALUES(?, ?, ?)", id,
38                 ↳ row["primaryTitle"], startYear)
39
40             # Insert genres
41             if row["genres"] != "\\N":
42                 for genre in row["genres"].split(","):
43                     db.execute("INSERT INTO genres
44                         ↳ (show_id, genre) VALUES(?, ?)",
45                         ↳ id, genre)

```

Program 8.12: import to multiple tables in SQL using python

```

1  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

```

1  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*