CS50 Quiz Preparation
Quiz Logistics
Quiz Logistics

- Released Sat 11/10 noon, due Thu 11/15 noon
- Open-book for all non-human resources
- Submit the quiz via submit50
How to Prepare

• Review lecture notes, source code, slides, video
• Attend or watch review session
• Take past years' tests and quizzes
• Review problem set specifications, sample solutions, distribution code
Themes
Themes

- Representing Data
- Abstraction
- Algorithms
- Trade-Offs
- Security
Week 0
Computational Thinking, Scratch
Binary

1

Bit
Binary

000111100

Byte
Binary

00011110

Byte
# ASCII

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>...</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>...</td>
<td>90</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>...</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>97</td>
<td>98</td>
<td>99</td>
<td>100</td>
<td>101</td>
<td>...</td>
<td>122</td>
</tr>
</tbody>
</table>
Input → Algorithms → Output
Functions
Variables

set my variable to 28
Conditions
Loops
Week 1
```c
#include <stdio.h>

int main(void)
{
    printf("hello, world!\n");
}
```
Types

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

string answer = get_string("What's your name?\n");
printf("%s\n", answer);
Conditions

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");
}
Loops

for (int i = 0; i < 10; i++)
{
    ...
}

while (x < 10)
{
    ...
    ...
    do
    {
        ...
    }
    while (x < 10);
Functions

```c
void cough(int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("cough\n");
    }
}
```
Floating-Point Imprecision

x: 2
y: 10
x / y = 0.2000000029802322387695312500
#include <stdio.h>
#include <unistd.h>

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

clang -o hello hello.c -lcs50
Debugging

• help50
• printf
• debug50
int value = 28;

int values[5];
Arrays

```c
int values[5];
values[0] = 10;
values[1] = 20;
values[3] = 40;
```

values

| 10 | 20 |   | 40 |
Strings

string name = "Zamyla";

name

Z a m y l a \0
Strings

string name = "Zamyla";

name

| 90 | 97 | 109 | 121 | 108 | 97 | 0 |
Command-Line Arguments

$ ./cash

$ make mario

$ clang -o hello hello.c
int main(int argc, string argv[]) {
    ...
}

Command-Line Arguments
Sorting

- Bubble Sort
- Selection Sort
- Merge Sort
## Time Complexity

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bubble Sort</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Selection Sort</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Merge Sort</td>
<td>$O(n \log n)$</td>
</tr>
</tbody>
</table>
Week 3
Memory
Pointers
Types

int  Integer
char  Character
Types

`int *`   Address of an Integer

`char *`  Address of a Character
Pointers

- `type *` declares a pointer that stores the address of a type
- `*x` takes pointer `x` and gets the value at the address
  - "dereference" operator
- `&x` takes variable `x` and gets its address
int a = 28;
int a = 28;
int a = 28;
int a = 28;
int a = 28;
int a = 28;
int b = 50;
int a = 28;
int b = 50;
int a = 28;
int b = 50;
int a = 28;
int b = 50;
int *c = &a;
int a = 28;
int b = 50;
int *c = &a;
int a = 28;
int b = 50;
int *c = &a;
int a = 28;
int b = 50;
int *c = &a;
*c = 14;
int a = 28;
int b = 50;
int *c = &a;
*c = 14;
int a = 28;
int b = 50;
int *c = &a;
*c = 14;
c = &b;
int a = 28;
int b = 50;
int *c = &a;
*c = 14;
c = &b;
int a = 28;
int b = 50;
int *c = &a;
*c = 14;
c = &b;
*c = 20;
int a = 28;
int b = 50;
int *c = &a;
*c = 14;
c = &b;
*c = 20;
Memory Layout
functions' variables and arguments
functions' variables and arguments
Stack

Heap
dynamically allocated memory

functions' variables and arguments
Stack

Heap
dynamically allocated memory

functions' variables and arguments
File I/O
File I/O

- Opening a file
- Reading from a file
- Writing to a file
- Changing position in a file
- Closing a file
FILE *foo = fopen("filename.txt", "w");

- Pointer to file
- Function to open file
- Name of file to open
- File mode (r, w, a)
fread(ptr, size, blocks, fp)

Where to store result of reading file

Size of a block to read

Number of blocks to read

Pointer to file to read from
fwrite(ptr, size, blocks, fp)

- Pointer to data to write
- Size of a block to write
- Number of blocks to write
- Pointer to file to write to
fseek(fp, offset, whence)

- Pointer to file
- How much to move file position by
- SEEK_SET,
  SEEK_CUR,
  SEEK_END
fclose(fp)

Pointer to file
Week 4
Data Structures
Arrays

10  20  30  40  50  60  70  80

90
typedef struct node
{
    int n;
    struct node *next;
}
node;
Linked List
Linked List

node *list = malloc(sizeof(node));
node *list = malloc(sizeof(node));
Linked List

node *list = malloc(sizeof(node));
Linked List

node *list = malloc(sizeof(node));
Linked List

```c
node *list = malloc(sizeof(node));
list->n = 10;
```
Linked List

node *list = malloc(sizeof(node));
list->n = 10;
Linked List

```c
node *list = malloc(sizeof(node));
list->n = 10;
list->next = NULL;
```
Linked List

```c
node *list = malloc(sizeof(node));
list->n = 10;
list->next = NULL;
```
Linked List

- Node at 0x500: n = 10, next
- Node at 0x600: n = 20, next
- Node at 0x700: n = 30
Linked List

0x500

n
10
next
0x600

0x600

n
20
next

0x700

n
30
next
Linked List

- Node 0x500:
  - Data: 10
  - Next: 0x600

- Node 0x600:
  - Data: 20
  - Next: 0x700

- Node 0x700:
  - Data: 30
  - Next: (null)
Linked List

0x500:

n
10
next
0x600

0x600:

n
20
next
0x700

0x700:

n
30
next
NULL
Linked List

```
0x500
n 10
next 0x600

0x600
n 20
next 0x700

0x700
n 30
next NULL
```
Linked List

0x500

n
10
next

0x600

n
20
next

0x700

n
30
next
Linked List

```
head

0x500
<table>
<thead>
<tr>
<th>n</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>next</td>
<td></td>
</tr>
</tbody>
</table>

0x600
<table>
<thead>
<tr>
<th>n</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>next</td>
<td></td>
</tr>
</tbody>
</table>

0x700
<table>
<thead>
<tr>
<th>n</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>next</td>
<td></td>
</tr>
</tbody>
</table>
```
Hash Tables
even_head

10 ➔ 28 ➔ 16 ➔ ...

odd_head

13 ➔ 5 ➔ 11 ➔ ...

act ➔ ace
bin ➔
cat ➔ cave
Hash Function

```c
unsigned int hash(const char *word) {
    return tolower(word[0]) - 'a';
}
```
Trees

![Tree Diagram]

- Node 1
- Node 2
- Node 3
- Node 4
- Node 5
- Node 6
- Node 7

The image shows a tree data structure with nodes 1, 2, 3, 4, 5, 6, and 7.
Binary Search Tree
Tries

A B C D ... X Y Z

A B C D ... X Y Z

A B C D ... X Y Z
Tries

typedef struct node
{
    bool is_word;
    struct node *children[27];
}
node;
Week 5
HTTP, HTML, CSS
TCP/IP
TCP/IP
TCP/IP

28.28.28.28

42.42.42.42
TCP/IP

From: 28.28.28.28
To: 42.42.42.42
Port: 80
Packet: 2 of 4
## DNS

<table>
<thead>
<tr>
<th>hostname</th>
<th>ip</th>
</tr>
</thead>
<tbody>
<tr>
<td>google.com</td>
<td>172.217.7.206</td>
</tr>
<tr>
<td>harvard.edu</td>
<td>23.22.75.102</td>
</tr>
<tr>
<td>yale.edu</td>
<td>104.16.243.4</td>
</tr>
<tr>
<td>apple.com</td>
<td>17.172.224.47</td>
</tr>
<tr>
<td>github.com</td>
<td>192.30.253.112</td>
</tr>
</tbody>
</table>
HTTP

GET / HTTP/1.1
Host: www.harvard.edu
HTTP Status Codes

- 200
- 301
- 403
- 404
- 500
- ...

HTML
<!DOCTYPE html>
<html>
<head>
  <title>hello!</title>
</head>
<body>
  hello, world!
</body>
</html>
CSS
CSS

```css
element {
    property: value;
}
```
#id
{
  property: value;
}

.class
{
    property: value;
}

JavaScript
> DOM

<!DOCTYPE html>
<html>
  <head>
    <title>hello!</title>
  </head>
  <body>
    hello, world!
  </body>
</html>
<!DOCTYPE html>
<html>
  <head>
    <title>hello!</title>
  </head>
  <body>
    hello, world!
  </body>
</html>
Week 6
Python
Variables

x = 28
y = "Hello"
z = True
print("Hello!")
print("Hello!", end=" ")
print(x)
print("Hello,", name)
if x > 0:
    print("Positive")
else:
    print("Not positive")
if x > 0:
    print("Positive")
elif x < 0:
    print("Negative")
else:
    print("Zero")
Lists

days = ['Sunday', 'Monday', 'Tuesday']
Loops

for i in range(5):
    print(i)
Loops

for item in items:
    print(item)
def add(a, b):
    return a + b
Data Structures

list

tuple

set

dict
Week 7
Web Programming
MVC

- Model
- View
- Controller
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html")
Week 8
SQL
Database

<table>
<thead>
<tr>
<th>Table</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>Table</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SQL Data Types

- **INTEGER**: smallint, integer, bigint
- **NUMERIC**: boolean, date, datetime, numeric(scale, precision), time, timestamp
- **REAL**: real, double precision
- **TEXT**: char(n), varchar(n), text
CREATE TABLE registrants (  id INTEGER PRIMARY KEY,  name VARCHAR(255),  dorm VARCHAR(255) )
SQL

INSERT INTO registrants (id, name, dorm)
VALUES (1, 'David', 'Matthews')
SELECT * FROM registrants
SELECT name, dorm FROM registrants
SELECT name, dorm FROM registrants
WHERE dorm = 'Matthews'
UPDATE registrants
SET name = 'David Malan'
WHERE id = 1
DELETE FROM registrants WHERE id = 1
Multiple Tables

• Foreign Keys
• Joining Tables
• Race Conditions
• SQL Injection Attacks
CS50 Quiz Preparation