Review for Test

October 12, 2016

Info

http://docs.cs50.net/2016/fall/test/about.html

- 72 hour window in which to take the test.
  - You should require much less than that.

- Released Fri 10/14 at noon, due via submit50 Mon 10/17 at noon.
  - Be sure to run `update50` in your IDE before submitting!
  - Submitting seven minutes late is equivalent to not submitting at all; don't wait until the last possible second.

Resources

- Consult the syllabus for a guide of topics.
  - We'll run through everything at a very high level today.
- Review lecture notes.
- Review lecture source code.
- Review lecture slides.
- (Re)watch lecture videos.
- Review problem set specifications, distribution code, and postmortems.

Resources

- Office hours
  - Today (Wednesday) at HSA through 5pm.
  - Tonight in Widener from 9pm – 11pm.
  - Thursday at HSA from 10am – 5:15pm.
  - Thursday night in SOCH from 9pm – 11pm.

Resources

- Office hours
  - Today (Wednesday) at HSA through 5pm.
  - Tonight in Widener from 9pm – 11pm.
  - Thursday at HSA from 10am – 5:15pm.
  - Thursday night in SOCH from 9pm – 11pm.

- No office hours during the Test (10/14 through 10/17).
- No office hours during Coding Contest (10/17 through 10/22).

Resources

- CS50 Discuss
  - You may post questions through Thu 10/13.
  - You may not post questions on Discuss from Fri 10/14 through Mon 10/17.
    - Staff will not respond to any questions during this time, but will monitor the forum.

Resources

- CS50 Discuss
  - You may post questions through Thu 10/13.
  - You may not post questions on Discuss from Fri 10/14 through Mon 10/17.
    - Staff will not respond to any questions during this time, but will monitor the forum.

- The only humans to which you may turn for help during the Test are the course's heads.

Resources

- CS50 Quiz Bank
  - Available at quizbank.cs50.net.
  - No login required.
  - Archive of old quiz questions from 2007—15, searching by topic and keyword.
  - Test this year will take a different format from years past, but the content being asked about does not differ widely.

Week 0

- Binary
  - Digits: 0, 1
  - Place values: 1s, 2s, 4s, 8s, 16s…

Week 0

- Binary
  - Digits: 0, 1
  - Place values: 1s, 2s, 4s, 8s, 16s…
- ASCII
  - Uniform standard for mapping of numbers to characters.
  - ‘A’ is 65, ‘a’ is 97…

Week 0

- Binary
  - Digits: 0, 1
  - Place values: 1s, 2s, 4s, 8s, 16s…
- ASCII
  - Uniform standard for mapping of numbers to characters.
  - ‘A’ is 65, ‘a’ is 97…
- Bytes
  - The value of a byte is context-dependent.
  - Maybe that 65 is just a 65.
  - In Microsoft Word that 65 might indeed be an ‘A’.
  - In Photoshop that 65 might represent the red value of a particular pixel.

Week 0

- Algorithms
  - Step by step sets of instructions for completing a task.
  - Counting the class.
  - Finding Mike Smith in a phone book.
  - Correctness versus efficiency.

Week 0

- Algorithms
  - Step by step sets of instructions for completing a task.
  - Counting the class.
  - Finding Mike Smith in a phone book.
  - Correctness versus efficiency.
- Scratch
  - Basic blocks – control, data, sound, looks.
  - Custom blocks – “functions”.
  - Events – when _____.

Week 1

- Loops
  - `for` – running a specific number of times.
  - `while` – running an unknown number of times, possibly zero.
  - `do-while` – running an unknown number of times, at least once.

Week 1

- Loops
  - `for` – running a specific number of times.
  - `while` – running an unknown number of times, possibly zero.
  - `do-while` – running an unknown number of times, at least once.
- Conditions
  - Boolean expressions – true or false
  - `if`, `else if`, `else`
  - `switch`
  - Ternary operator - `?:`

Week 1

- Variables
  - Containers that hold information.
  - Before using, need to declare.
  - Variables hold information of a specific type, and have a name.
  - Use = to assign values to variables, right-to-left.

Week 1

- Variables
  - Containers that hold information.
  - Before using, need to declare.
  - Variables hold information of a specific type, and have a name.
  - Use `=` to assign values to variables, right-to-left.
- Compiling
  - **`make`** is a utility we use to turn our C code into executable programs.
  - **`clang`** is a compiler that does the hard work of this translation.
  - Computers only understand machine code, not our C source.
  - Preprocessing, compiling, assembling, linking.

Week 1

- Data Types
  - Native data types in C
    - `int`, `char`, `float`, `double`, `long long`
  - Additional data types
    - `bool`, `string`
  - `signed` and `unsigned`
  - 1 byte
    - `bool`, `char`
  - 4 bytes
    - `float`, `int`
  - 8 bytes
    - `double`, `long long`, `string`

Week 1

- Overflow
  - With an integer, we only have 4 bytes (32 bits) to work with.
  - We can’t store any number equal to or greater than 2^{32}.

Week 1

- Overflow
  - With an integer, we only have 4 bytes (32 bits) to work with.
  - We can’t store any number equal to or greater than 2^{32}.
- Imprecision
  - With a float, we only have 4 bytes (32 bits) to work with.
  - We cannot possibly represent every real number.

Week 2

- Bugs
  - Implicit declaration of functions.
  - Use of undeclared identifier.
  - Out of bounds error.
  - Segmentation fault.
  - `help50`, `debug50`.

Week 2

- Bugs
  - Implicit declaration of functions.
  - Use of undeclared identifier.
  - Out of bounds error.
  - Segmentation fault.
  - `help50`, `debug50`.
- Cryptography
  - Art and science of obscuring information.
  - Rotational cipher.

Week 2

- Strings
  - A sequence of characters.

Week 2

- Strings
  - ~~A sequence of characters.~~
  - An array of characters.
  - Length of a string is available via the function `strlen`.
  - Each character of the string is available with `str[i]`
    - `0 <= i < strlen(str)`
  - All strings end with the `\0` character.

Week 2

- Strings
  - ~~A sequence of characters.~~
  - An array of characters.
  - Length of a string is available via the function **`strlen`**.
  - Each character of the string is available with **`str[i]`**
    - **`0 <= i < strlen(str)`**
  - All strings end with the **`\0`** character.
- Typecasting
  - Think back to ASCII, every character is associated with a number.
  - We can treat characters as numbers and do math with them using their ASCII values.
  - Explicit typecasting uses a **`(type)`** specifier.

Week 2

- Reference Tools
  - Manual pages are part of most Linux installations.
  - `reference.cs50.net` is written by the staff.
  - Many online equivalents for C and other languages.

Week 2

- Reference Tools
  - Manual pages are part of most Linux installations.
  - `reference.cs50.net` is written by the staff.
  - Many online equivalents for C and other languages.
- Command-Line Arguments
  - By modifying our prototype for `main`, the user can supply extra information to our programs at runtime.
  - `int main(int argc, string argv[])`
  - `argc` refers to how many things the user typed.
  - `argv` is an array of strings storing what they actually typed.

Week 3

- Searching
  - Linear search considers a general array, and looks over each element from beginning to end until it finds the target.
  - Binary search considers a **sorted** array, looks at the middle, and discards half of the remaining array until it finds the target.

Week 3

- Searching
  - Linear search considers a general array, and looks over each element from beginning to end until it finds the target.
  - Binary search considers a **sorted** array, looks at the middle, and discards half of the remaining array until it finds the target.
- Sorting
  - Selection sort: Find the smallest remaining, swap with the first.
  - Bubble sort: Adjacent pairs out of order? Swap them.
  - Insertion sort: Shift previously sorted elements to make room.
  - Merge sort: Sort partial arrays, then combine them together.

Week 3

- Big O
  - Provides us with a shorthand way to refer to the running time of various algorithms.
  - In CS50, normally O describes the *upper bound* on runtime.
  - In CS50, normally Ω describes the *lower bound* on runtime.

Week 3

Algorithm	Upper bound (O)	Lower bound (Ω)
Linear search	n	1
Binary search	$\log n$	1
Selection sort	n^2	n^2
Bubble sort	n^2	n
Insertion sort	n^2	n
Merge sort	$n \log n$	$n \log n$

Week 3

- Recursion
  - Problem solving technique where we use the solution to a smaller problem to inform the solution to a larger one.
  - Series summation, factorial, exponentiation, Fibonacci sequence…
  - A recursive algorithm has two parts:
    - Base case – recursion stops; the simple case we have a solution for
    - Recursive case – recursion continues; make a more complex case a little bit simpler, tending towards the base case.

Week 3

```
int fact(int n)
{
    if (n <= 0)
        return 1;
    return n * fact(n-1);
}
```

```
int fact(int n)
{
    int product = 1;
    while (n > 0)
    {
        product *= n--;
    }
    return product;
}
```

Week 4

- Call Stack
  - Swapping values in a separate function has no effect in the *calling* function.
  - Passing variables to a function gives that function its own local copy of those variables; our original ones remain intact.
  - A function call creates a stack frame.
  - The most recently called function is the one with the “highest” frame on the stack, and is the only function active.
  - All other functions are “on pause” where they left off.

Week 4

- Pointers
  - How can we access memory in other functions?
  - Pointers are addresses, specifically the addresses of variables we care about.
  - Finding a variable's address: &
  - Going to an address to manipulate a variable: *
    - Dereferencing

Week 4

```
int main(void)
{
    int x = 4;
    int *px = &x;
    *px = 5;
    printf("%i\n", x);
}
```

Week 4

```c
int main(void)
{
    int x = 4;
    change(x);
    printf("%i\n", x);
}
```

```c
void change(int x)
{
    x = 5;
    return;
}
```

Week 4

```c
int main(void)
{
    int x = 4;
    change(&x);
    printf("%i\n", x);
}
```

```c
void change(int *x)
{
    *x = 5;
    return;
}
```

Week 4

- Strings Redux
  - The variable name of a string is behind the scenes just a pointer to (aka the address of) its first character.
  - `string s = "CS50";`
  - `string t = "CS50";`

Week 4

- Strings Redux
  - The variable name of a string is behind the scenes just a pointer to (aka the address of) its first character.
  - `string s = get_string(); // user types "CS50"`
  - `string t = get_string(); // user types "CS50"`
- Dynamic Memory
  - If I need memory while my program is running that I didn't anticipate at compile-time, I can use **`malloc`**.
  - **`malloc`** expects a number of bytes as a parameter, and gives you back a pointer.
    - **`sizeof`** is helpful here!
  - Need to **`free`** all dynamically allocated memory.

Week 4

- Valgrind
  - Tool that we can use to spot memory leaks in our program.
  - Tells about any invalid thing we do with program's memory.

Week 4

- Valgrind
  - Tool that we can use to spot memory leaks in our program.
  - Tells about any invalid thing we do with program's memory.
- Buffer Overflow
  - Integer overflow occurs when we try and store an integer larger than we are capable of storing.
  - Buffer overflow occurs when we try and store a string larger than we've set aside space for.
  - Can be used to malicious effect.

Week 4

- Structures
  - C permits us to *encapsulate* data, by wrapping it up into a structure.
  - Group together related data into a single entity.
  - Dot operator to access a structure's members.
  - If we have pointers to structures, we use arrow (`->`) instead of dot, to dereference the pointer, then access the member.
  - `typedef` to give us cleaner type names.

Week 4

```
struct student
{
    char name[20];
    char house[20];
    int year;
    float gpa;
}
```

Week 4

```
struct student
{
    char name[20];
    char house[20];
    int year;
    float gpa;
}
```

```
struct student maria;

strcpy(maria.name, "Maria");
strcpy(maria.house, "Cabot");
maria.year = 2018;
maria.gpa = 4.00;
```

Week 5

- Linked Lists
  - Arrays suffer from a fixed-size limitation.
  - Lists grow and shrink with ease, but require dynamic memory.
  - Structure (node) with at least two members:
    - Data
    - A pointer to another structure in the same linked list (or to `NULL`).
  - Insertion and deletion can be constant time, O(1), operations.
  - Lookup/search is O(n), since we lose random access.
    - Start at the beginning of the chain, and work your way to the end.
    - Linear search.

Week 5

- Stacks
  - LIFO (last in, first out)
  - Linked list: You can only ever insert or delete from the head of the list.
  - Array: Keep track of most recently added element at all times.

Week 5

- Stacks
  - LIFO (last in, first out)
  - Linked list: You can only ever insert or delete from the head of the list.
  - Array: Keep track of most recently added element at all times.
- Queues
  - FIFO (first in, first out)
  - Linked list: You can only ever insert at the head of the list and delete from the tail of the list.
  - Array: Keep track of number of elements and “oldest” element at all times.

Week 5

- Trees
  - Node with normally at least three members:
    - Data
    - At least two pointers to other nodes lower in the tree (or to NULL)
  - Binary trees
  - Binary search trees
  - Tries
  - Insertion and deletion can be constant time, O(1), operations.
  - Lookup/search is **O(log n)** in a binary search tree.
  - Lookup/search is **O(1)** in a trie.

Week 5

- Hash tables
  - Combination of a linked list and an array.
  - Use a hash function to get a value for your data.
  - Store in the linked list located at that index of the array.