CS50 Section. Week 6. Quiz 0 Review. 10/13/15.

Tuesdays 7:00-8:30pm, Science Center 309A
https://github.com/hathix/cs50-section

Neel Mehta
neelmehta@college.harvard.edu
(215) 990-6434

Get these handouts at https://github.com/hathix/cs50-section/tree/master/handouts.

Practice problems

odd (Fall 2014 #17)

Complete the implementation of `odd` below in such a way that the function returns `true` if `n` is odd and `false` if `n` is even.

```
bool odd(unsigned int n)
{




}
```

Solution: http://cdn.cs50.net/2014/fall/quizzes/0/key0.pdf

GetRandom (Fall 2011 #13)

C comes with `rand`, which returns a pseudorandom `int` specifically between `0` and `RAND_MAX`, inclusive, where `RAND_MAX` is a large, positive constant. If only there were a way to generate a pseudorandom number in C between any two values! Complete the implementation of `GetRandom` below in such a way that it uses `rand` but returns a pseudorandom `int` between `min` and `max`, inclusive. You may assume that both `min` and `max` will be non-negative and less than or equal to `RAND_MAX`. And you may assume that `max` will be greater than or equal to `min`. You needn't worry about seeding. And no need to `#include` any header files (e.g. `stdlib.h`, in which `rand` is declared.)

```
int GetRandom(int min, int max)
{
    int n = rand();









}
```

Solution: http://cdn.cs50.net/2011/fall/quizzes/0/key0.pdf

strlen (Fall 2013 #10)

Suppose that you've forgotten which header file declares `strlen`, and so you need to re-implement it yourself. Bah. Even worse, neither `[` nor `]` currently works on your keyboard (or pencil or pen). Without calling any functions at all and without using any square brackets, complete the implementation of `strlen` below **using pointer arithmetic** in such a way that the function returns the length of `s`. If `s` happens to be `NULL`, your implementation of `strlen` should return 0.

```
int strlen(char* s)
{









}
```

Solution: http://cdn.cs50.net/2013/fall/quizzes/0/key0.pdf

Pointers (Fall 2010 #29)

Consider the code below.

```
int i = 1;
int j = 2;
int k = 7;
int* p = &j;
*p += 1;
```

Complete the table below, specifying in the right-hand column exactly what would be printed by each call to `printf`, assuming `printf` is called after *all* of the lines above have been executed. As an example, we've filled in the first box for you.

statement	what would be printed
`printf("%d", i)`	1
`printf("%d", j)`	
`printf("%d", *p)`	
`printf("%d", k)`	

Solution: http://cdn.cs50.net/2010/fall/quizzes/0/key0.pdf

More Pointers (Fall 2012 #28)

Recall the program below, which crashed when Binky tried to execute it.

```
#include <stdlib.h>

int main(void)
{
    //
    //
    int* x;

    //
    //
    int* y;

    //
    //
    x = malloc(sizeof(int));

    //
    //
    *x = 42;

    //
    //
    *y = 13;

    //
    //
    y = x;

    //
    //
    *y = 13;

    return 0;
}
```

Atop some of this program's lines are placeholders for comments. Next to each such `//`, explain in precise (but succinct) technical terms what the line immediately below it is doing. Write on the program itself.

Solution: http://cdn.cs50.net/2012/fall/quizzes/0/key0.pdf

Stuff for your cheat sheet

Algorithms

Name	What it does	Worst-case	Best-case
Linear search	Finds an element in a list by searching left-to-right	O(n)	Ω(1)
Binary search	Finds an element in a sorted list using divide-and-conquer	O(log n)	Ω(1)
Bubble sort	Sorts a list by bubbling biggest elements to end	O(n^2)	Ω(n)
Selection sort	Sorts a list by moving smallest elements to front	O(n^2)	Ω(n^2)
Insertion sort	Sorts a list by moving elements to properly sorted place	O(n^2)	Ω(n)
Merge sort	Recursively sorts a list by partitioning and merging	O(n log n)	Ω(n log n)

More at http://www.bigocheatsheet.com/.

Data types

The below are for a 64-bit machine (so named because pointers are 64 bits) like the CS50 IDE.

Type	char	int	float	double	long long	int*
Size (bytes)	1	4	4	8	8	8

- Unsigned types are the same size as the normal types (e.g. `unsigned int`, like `int`, is 4 bytes)
- All pointers are the same size (e.g. `double*`, like `int*`, is 8 bytes)

Useful functions

Character Class Tests <ctype.h>

alphanumeric?	`isalnum(c)`
alphabetic?	`isalpha(c)`
control character?	`iscntrl(c)`
decimal digit?	`isdigit(c)`
printing character (not incl space)?	`isgraph(c)`
lower case letter?	`islower(c)`
printing character (incl space)?	`isprint(c)`
printing char except space, letter, digit?	`ispunct(c)`
space, formfeed, newline, cr, tab, vtab?	`isspace(c)`
upper case letter?	`isupper(c)`
hexadecimal digit?	`isxdigit(c)`
convert to lower case	`tolower(c)`
convert to upper case	`toupper(c)`

String Operations <string.h>

`s` is a string; `cs`, `ct` are constant strings

length of `s`	`strlen(s)`
copy `ct` to `s`	`strcpy(s,ct)`
concatenate `ct` after `s`	`strcat(s,ct)`
compare `cs` to `ct`	`strcmp(cs,ct)`
only first `n` chars	`strncmp(cs,ct,n)`
pointer to first `c` in `cs`	`strchr(cs,c)`
pointer to last `c` in `cs`	`strrchr(cs,c)`
copy `n` chars from `ct` to `s`	`memcpy(s,ct,n)`
copy `n` chars from `ct` to `s` (may overlap)	`memmove(s,ct,n)`
compare `n` chars of `cs` with `ct`	`memcmp(cs,ct,n)`
pointer to first `c` in first `n` chars of `cs`	`memchr(cs,c,n)`
put `c` into first `n` chars of `s`	`memset(s,c,n)`

Input/Output <stdio.h>

Standard I/O

standard input stream	`stdin`
standard output stream	`stdout`
standard error stream	`stderr`
end of file (type is `int`)	`EOF`
get a character	`getchar()`
print a character	`putchar(`*chr*`)`
print formatted data	`printf("`*format*`",`arg_1`,...)`
print to string `s`	`sprintf(s,"`*format*`",`arg_1`,...)`
read formatted data	`scanf("`*format*`",&`$name_1$`,...)`
read from string `s`	`sscanf(s,"`*format*`",&`$name_1$`,...)`
print string `s`	`puts(s)`

File I/O

declare file pointer	`FILE *`*fp*`;`
pointer to named file	`fopen("`*name*`","`*mode*`")`
modes: `r` (read), `w` (write), `a` (append), `b` (binary)	
get a character	`getc(`*fp*`)`
write a character	`putc(`*chr*`,`*fp*`)`
write to file	`fprintf(`*fp*`,"`*format*`",`arg_1`,...)`
read from file	`fscanf(`*fp*`,"`*format*`",`arg_1`,...)`
read and store `n` elts to `*ptr`	`fread(*ptr,eltsize,n,`*fp*`)`
write `n` elts from `*ptr` to file	`fwrite(*ptr,eltsize,n,`*fp*`)`
close file	`fclose(`*fp*`)`
non-zero if error	`ferror(`*fp*`)`
non-zero if already reached EOF	`feof(`*fp*`)`
read line to string `s` (< `max` chars)	`fgets(s,max,`*fp*`)`
write string `s`	`fputs(s,`*fp*`)`

Conversions stdlib.h

convert string `s` to double	`atof(s)`
convert string `s` to integer	`atoi(s)`
convert string `s` to long	`atol(s)`

Common errors

Error	Why it happened
undefined reference to 'function'	You forgot to call the linker with, e.g., `-lcs50`
implicitly declaring library function	You forgot to `#include` the `.h` file
implicit declaration of function 'function' is invalid	You forgot to `#include` the `.h` file
more '%' conversions than data arguments	You called, e.g., `printf("%i and %i \n", 5)` (you passed 1 number but should have given 2)
definitely lost: # bytes in # blocks	You forgot to `free` memory you allocated with `malloc`
invalid write of size #	You tried changing a value beyond the end of an array
use of undeclared identifier	You forgot to declare the variable, or it's out of scope

File I/O

Opening and closing files

https://reference.cs50.net/stdio.h/fopen, https://reference.cs50.net/stdio.h/fclose

```
FILE* file_pointer = fopen(filename, "r");
// do stuff
fclose(file_pointer);
```

Reading from files

https://reference.cs50.net/stdio.h/fread

Reading into an array:

```
int length = 50;
int destination[length];
fread(destination, sizeof(int), length, file_pointer);
```

Reading into a single variable:

```
int destination;
fread(&destination, sizeof(int), 1, file_pointer);
```

Writing to files

https://reference.cs50.net/stdio.h/fwrite

Writing from an array:

```
int length = 50;
int source[length];
fwrite(source, sizeof(int), length, file_pointer);
```

Writing from a single variable:

```
int source;
fwrite(&source, sizeof(int), 1, file_pointer);
```

Moving file pointer

https://reference.cs50.net/stdio.h/fseek

```
int distance = 50;
fseek(file_pointer, distance, SEEK_CUR);
```