Computer Science 50: Introduction to Computer Science I
Harvard College
Fall 2008

Quiz 1

out of 63 points

Print your name on the line below.

____

Do not turn this page over until told by the staff to do so.

This quiz is "closed-book." However, you may utilize during the quiz one two-sided page (8.5" × 11") of notes, typed or written, and a pen or pencil, but nothing else.

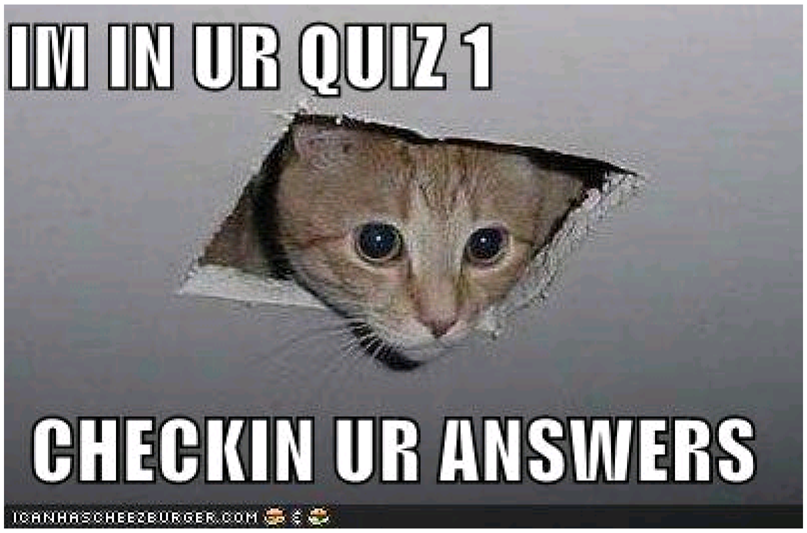
Scrap paper is included at this document's end.

Please circle your section leader's name.

Aaron Oehlschlaeger	Kent Rakip
Alex Chang	Linfeng Yang
Andrew Sellergren	Mike Tucker
Andy Lei	Patrick Quinn
Batool Ali	Patrick Schmid
Brett Thomas	Peter Bailis
Chris Power	Peter Lifland
Chris Simmons	Sanjay Gandhi
Daniel Carroll	Tom Wionzek
Daniel Steinbrook	Tomo Lazovich
David Ramos	Vivek Sant
David Wu	Will Phan
Jesse Cohen	Yuhki Yamashita
Josh Bolduc	Zeina Oweis
Ken Parreno	

for staff use only

final score out of 63



***O*(hai).**

0. (6 points.) For each of the operations below, provide upper and lower bounds on its running time. Assume that *n* represents the number of integers already in each data structure.

	O	**Ω**
inserting an integer into a sorted doubly linked list		
inserting an integer into a sorted singly linked list		
sorting an unsorted array of integers with heapsort		

for staff use only
–

Bits.

1. (2 points.) Perform the bitwise operation below. Recall that | denotes OR.

```
  10111111
| 00000001
----------
```

2. (2 points.) Perform the bitwise operation below. Recall that ^ denotes XOR.

```
  10111111
^ 00000001
----------
```

3. (1 point.) Consider the C code below.

```
printf("%d", n << 1);
```

If `n` happens to be `25`, what number gets printed?

for staff use only

–

4. (1 point.) Consider the C function below.

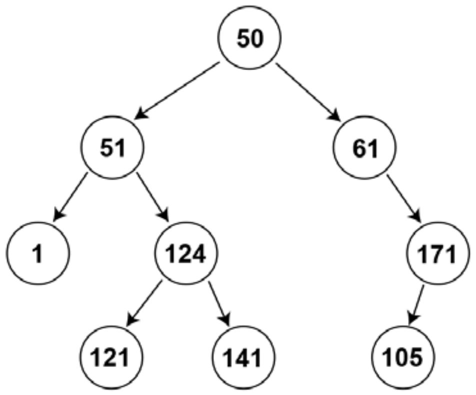
```
bool
f(int n)
{
    if (n & 1)
        return true;
    else
        return false;
}
```

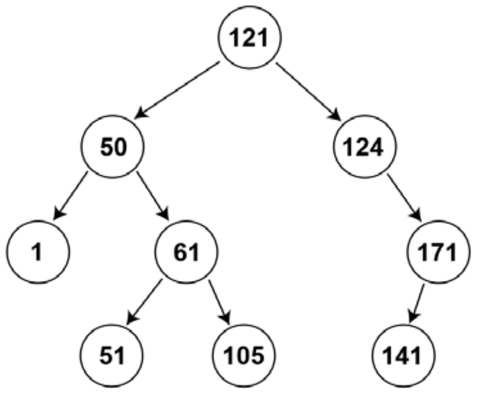
What does this function return if `n` is an odd integer?

Trees.

Recall that a *tree* is a data structure in which each node has some value along with zero or more children. A *binary tree*, meanwhile, is a tree in which each node has at most two children. A *binary search tree*, finally, is a binary tree, the value of whose root is (1) greater than that of its left child, if any, and any descendants thereof and (2) less than that of its right child, if any, and any descendants thereof. Moreover, each child is itself the root of a binary search tree. To be clear, a binary search tree is necessarily a binary tree, but a binary tree is not necessarily a binary search tree. For simplicity, assume that trees' values are integers and that trees may not contain duplicate values.

5. (1 point.) Which of the trees below is a binary search tree, the one on the left or the one on the right? Circle your answer.





for staff use only

–

6. (2 points.) Assume that, when implemented in C, the value of each node in a binary tree is an `int`. Complete the below definition of `node`.

```
typedef struct node
{







}
node;
```

7. (4 points.) Suppose that a C function called `find` is supposed to return `true` if any node in a binary tree contains a particular value, else it is supposed to return `false`. Complete the below definition of `find`, based on your definition of `node`. Do not assume that `tree` is non-`NULL` or that `tree` is a binary search tree.

```
bool
find(node *tree, int n)
{
```

for staff use only

–

8. (4 points.) Suppose that a C function called `insert` is supposed to insert a new node with some value into a binary search tree so long as that value is not already present in some other node. If a new node with that value is ultimately inserted, this function is supposed to return `true`, else it is supposed to return `false`. Complete the below definition of `insert`, based on your definition of `node`. Do not assume that `tree` is non-`NULL`, but you may assume that `tree`, if non-`NULL`, is a binary search tree. Be sure that `tree` remains a binary search tree after any insertion, but do not worry about keeping its height balanced.

```
bool
insert(node *tree, int n)
{
```

for staff use only

–

Rapid Fire.

Answer each of the questions below in one sentence.

9. (0 points.) Is it Christmas?

10. (2 points.) What does it mean if a machine is little-endian?

11. (2 points.) When are SQL injection attacks possible?

12. (2 points.) What is one property of a good hash function?

13. (2 points.) For what purpose are regular expressions useful?

14. (0 points.) Has the Large Hadron Collider destroyed the world yet?*

for staff use only

–

* Maybe by 2:30 P.M.?

Alert!

15. (2 points.) Recall that you can register DOM listeners (*i.e.*, event handlers) using XHTML attributes like `onclick`, as in the below.

```
<input id="go" onclick="alert('Click!');" type="button" value="Go" />
```

But you can also register the same programmatically without using that attribute, as in the below.

```
document.getElementById("go").onclick = function() { alert('Click!'); };
```

Or you can use APIs like Google's to accomplish the same:

```
GEvent.addDomListener(document.getElementById("go"), "click", function() {
    alert('Click!');
});
```

Explain why the below, though, does not accomplish the same, making clear what it actually does:

```
document.getElementById("go").onclick = alert('Click!');
```

Bosco.

16. (2 points.) Suppose that Lord Dark Helmet wants to withdraw `$amount` from his bank account, and so his bank's ATM (automated teller machine) executes the below PHP code, whereby `dispense` is a function that causes cash to come out.

```
$result = mysql_query("SELECT cash FROM accounts WHERE customer='dhelmet'");
$row = mysql_fetch_array($result);
if ($amount > 0 && $amount < $row["cash"])
{
    dispense($amount);
    mysql_query("UPDATE accounts SET cash = cash - $amount WHERE customer='dhelmet'");
}
```

Explain how this code, perhaps over the course of multiple withdrawals, might allow Lord Dark Helmet to withdraw more money than he actually deposited.[†]

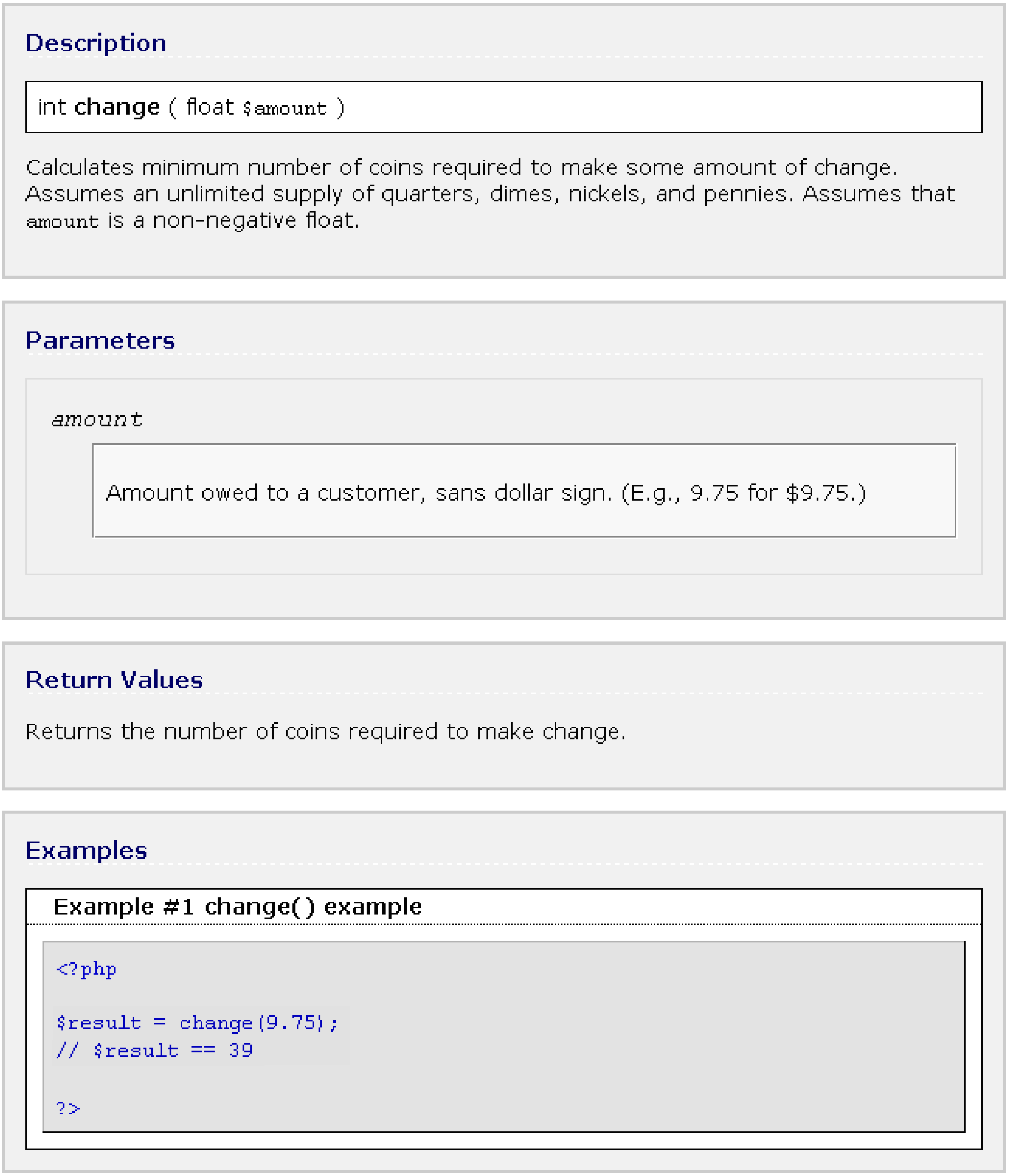
[†] And, no, it's not an interest-bearing account. Nice try.

for staff use only

–

Time for Change.

17. (4 points.) Consider the documentation below for an imaginary PHP function.



Description

int **change** (float `$amount`)

Calculates minimum number of coins required to make some amount of change. Assumes an unlimited supply of quarters, dimes, nickels, and pennies. Assumes that `amount` is a non-negative float.

Parameters

`amount`

Amount owed to a customer, sans dollar sign. (E.g., 9.75 for $9.75.)

Return Values

Returns the number of coins required to make change.

Examples

Example #1 change() example

```php
<?php

$result = change(9.75);
// $result == 39

?>
```

Based on its documentation, complete the implementation of this function below.

```
function change($amount)
{
```

for staff use only

–

What are you looking 404?

18. (4 points.) Consider the HTTP headers below, submitted by a browser upon submission of some form.

```
GET /index.php?from=kitten HTTP/1.1
Host: speaklolcat.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

In the space below, complete the implementation of an XHTML form whose submission could have generated these headers.

```
<form
```

for staff use only

–

O hai again, C.

19. (4 points.) Recall that C code becomes executable by way of a sequence of steps: pre-processing, compiling, assembling, and linking. Consider the source code below.

```
#include <stdio.h>

int
main(int argc, char *argv[])
{
    printf("Help, I'm trapped in a binary!\n");
}
```

With respect to this code, explain briefly what happens in each of those steps by completing the sentences below.

During pre-processing...

During compiling...

During assembling...

During linking...

for staff use only

–

Design Decisions.

For each pair below, *x* versus *y*, argue in one sentence when you should use *x* over *y* (or, if you prefer, *y* over *x*).

20. (2 points.) *C* versus *PHP*

21. (2 points.) *array* versus *linked list*

22 (2 points.) *CSV* versus *XML*

23. (2 points.) *decimal* versus *hexadecimal*

24. (2 points.) *GET* versus *POST*

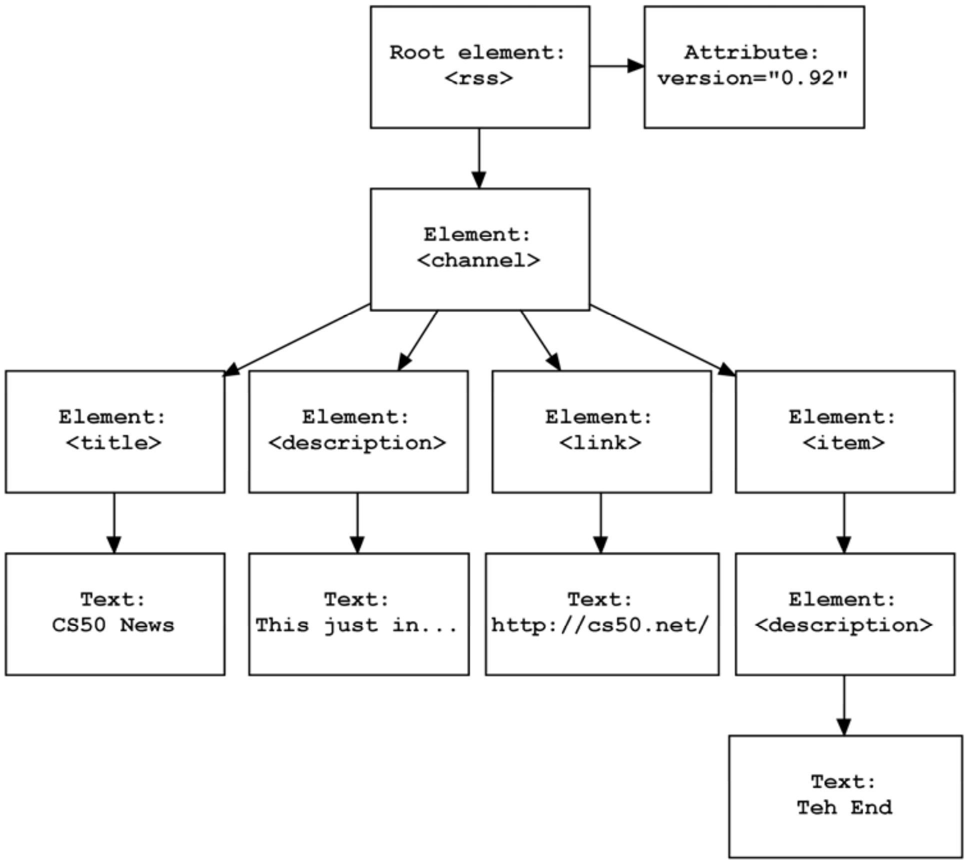
25. (2 points.) *vertical scaling* versus *horizontal scaling*

for staff use only

–

DOM, DOM DOM DOM, DOM.

26. (4 points.) Consider the (simplified) DOM below for an RSS feed.



In the space below, write the actual RSS (*i.e.*, XML) implied by this DOM.

for staff use only

–

Scrap Paper.

Nothing on this page will be examined by the staff unless otherwise directed in the space provided for some question.