This is **CS 50**.
Harvard College Fall 2009

Quiz 0

out of 55 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, nothing else.

Scrap paper is included toward this document's end.
Unless otherwise noted, assume that any problems herein refer to C.

Please circle your section leader's name.

Andrew Sellergren	Ken Parreno
Batool Ali	Kent Rakip
Dan Nevius	Lee Evangelakos
David Robinson	Marta Bralic
Derek Lietz	Matthew Chartier
Doug Lloyd	Michelle Konstadt
Drew Robb	Mike Teodorescu
Filip Zembowicz	Mike Tucker
Jesse Cohen	Nathan Leiby
John Selig	Patrick Quinn
Jon Noronha	Rose Cao
Jonathan Yip	Saba Zaidi
Josh Bolduc	Ted Rogers
Karim Atiyeh + Thomas Prufer	Yuhki Yamashita

for staff use only

final score out of 55

Multiple Choice.

For each of the following questions or statements, circle the letter (a, b, c, or d) of the one response that best answers the question or completes the statement; you need not explain your answers.

0. (0 points.) Which is Happy Cat?

a.

b.

c.

d.

1. (1 point.) What is the running time of binary search on a (sorted) phonebook with *n* pages?
   a. $O(1)$
   b. $O(\log n)$
   c. $O(n)$
   d. $O(n^2)$

2. (1 point.) How many bits are in a byte?
   a. 2
   b. 8
   c. 16
   d. 32

3. (0 points.) Gur nafjre vf p.
   a. n
   b. o
   c. p
   d. q

for staff use only

–

4. (1 point.) If the value of a `char`, `c`, is `'0'`, which line of code converts `c` to an `int`, `n`, whose integral value is `0`?
   a. `int n = (int) c;`
   b. `int n = atoi(c);`
   c. `int n = (int) c – 'A';`
   d. `int n = (int) c – '0';`

True or False.

For each of the statements below, circle T if the statement is true or F if the statement is false.

5.	T	F	(1 point.) `make` is a compiler.
6.	T	F	(1 point.) All `.c` files must have a `main` function.
7.	T	F	(1 point.) Double ROT13 is less secure than triple DES.
8.	T	T	(0 points.) I wish I lived in Mather House.

Short Answers.

9. (2 points.) Recall the implementation of Bubble Sort from lecture, below.

```
Repeat n times:
    For each element i:
        If element i and its neighbor are out of order:
            Swap them.
```

Even though we know Bubble Sort to be in $\Omega(n)$, this particular implementation is in $\Omega(n^2)$. In the space below, re-implement Bubble Sort in pseudocode in such a way that your version is in $\Omega(n)$. Any form of pseudocode suffices; you need not mimic our style or syntax.

for staff use only

–

10. (2 points.) If global variables are accessible by all functions anyway, why not declare all variables as global rather than use local variables at all?

11. (2 points.) Recall the algorithm from lecture below.

```
1. Stand up.
2. Think to yourself "I am #1."[1]
3. Pair off with someone standing, add your numbers together, and adopt
   the sum as your new number.
4. One of you should sit down, the other should go back to step 3.
```

Explain why this algorithm, if *n* students execute it together, is in *O*(log *n*).

12. (2 points.) Recall that the formula for conversion from Celsius, C, to Fahrenheit, F, is:

$C = (5/9) \times (F - 32)$

Consider the implementation of this formula in C (the language, not the temperature scale) below:

```
float c = (5/9) * (f - 32);
```

No matter the value of `f`, this code always assigns `c` a value of `0.0`, even if the value of `f`, a `float`, is a much warmer temperature (e.g., `212.0`). In no more than two sentences, explain why.

Re-implement the formula correctly below with just one line of code. Assume that `f` has already been declared as a `float`. We'll get you started:

```
float c =
```

[1] Not after this quiz![*]
[*] jk!

for staff use only

–

13. (2 points.) Recall that CS50's library defines `GetInt`, a function that gets an `int` from a user, pestering him or her to retry if the user fails to provide only an `int` (surrounded, perhaps, by whitespace). Recall the actual implementation of `GetInt`, below, whose comments (so sadly) did not make it to press.

```
int
GetInt()
{
    while (true)
    {
        string line = GetString();
        if (line == NULL)
            return INT_MAX;

        int n; char c;
        if (sscanf(line, " %d %c", &n, &c) == 1)
        {
            free(line);
            return n;
        }
        else
        {
            free(line);
            printf("Retry: ");
        }
    }
}
```

Explain precisely how this function detects whether a user has provided only an `int` (surrounded by nothing other than, perhaps, whitespace). Put another way, explain precisely how this function detects whether a user has provided something other than an `int`.

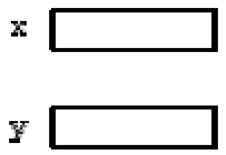
for staff use only

–

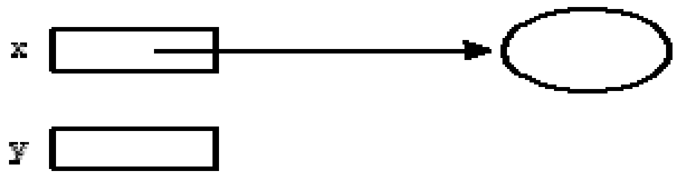
Pointer Fun with Binky.[2]

14. (3 points.) Below are three lines of code, alongside which are depictions thereof. You may recall that Binky drew these same pictures with clay.

```
int *x;
int *y;
```

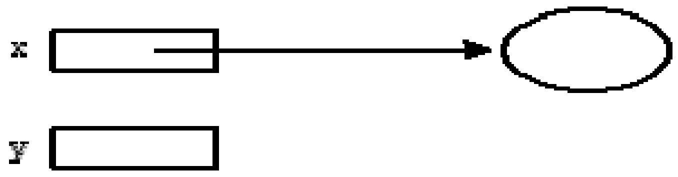


```
x = malloc(sizeof(int));
```

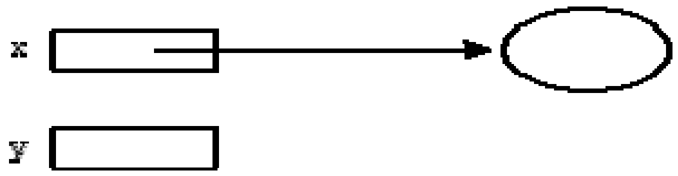


Perhaps needless to say, the oval represents an `int` on the heap, and each rectangle represents a pointer (not on the heap). Below are three more lines of code whose depictions are not yet complete. Complete the depictions by drawing arrows and/or numbers as the code dictates.

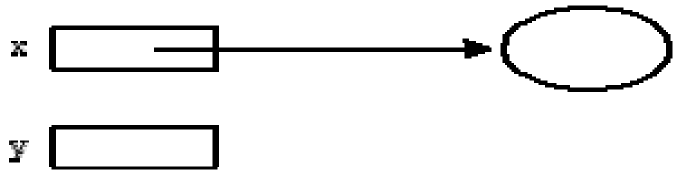
```
*x = 42;
```



```
y = x;
```



```
*y = 13;
```

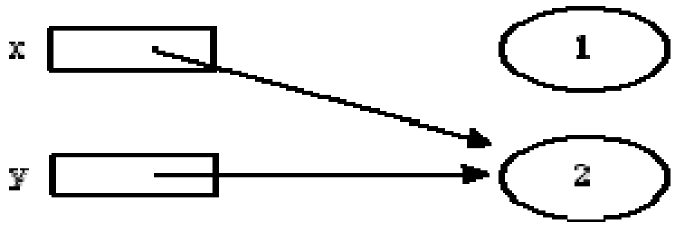


[2] Excerpted from document 106 in the Stanford CS Education Library. This and other free materials are available at `cslibrary.stanford.edu`.

for staff use only

–

15. (3 points.) Now we've a picture but no code! In the space at left below, write a few lines of code that collectively create this picture in memory. As before, each oval represents an `int` on the heap, and each rectangle represents a pointer (not on the heap).



sizeof.

16. (3 points.) Recall that `sizeof` returns the size, in bytes, of a data type. Complete the table below, one of whose rows we've plucked off for you. Assume a 32-bit x86 architecture like `nice.fas.harvard.edu`.

`sizeof(char)`	
`sizeof(char *)`	
`sizeof(int)`	4
`sizeof(int *)`	
`sizeof(long)`	
`sizeof(long long)`	
`sizeof(float)`	

for staff use only

–

Rapid Fire.

Answer the questions below in no more than three sentences each.[3]

17. (1 point.) Why do we say that `GetString`, defined in CS 50's library, can induce memory leaks?

18. (1 point.) Why ever use `switch`, given that you can implement the same functionality using `if`, `else if`, and `else`?

19. (1 point.) Why is Vigenère's cipher said to be stronger than Caesar's?

20. (1 point.) How can buffer overflow attacks be prevented?

[3] Don't make us count!

for staff use only

–

Swap Time.

21. (4 points.) Consider the definition of a `student` below.

```
typedef struct student
{
    char *name;
    char *house;
}
student;
```

Suppose that some student's been Quaded and (obviously) wants to transfer to Mather. The only way to do so, though, is to trade rooms with a Matherite. Let's tell this same story in code:

```
student a;
a.name = "Jon";
a.house = "Mather";

student b;
b.name = "Mike";
b.house = "Currier";

swap(&a, &b);

printf("%s now lives in %s.\n", a.name, a.house);
printf("%s now lives in %s.\n", b.name, b.house);
```

Mike desperately hopes the above prints these strings:

```
Jon now lives in Currier.
Mike now lives in Mather.
```

But he needs you to implement `swap`! Complete its implementation below. Take care not to segfault or to send anyone to Dunster.

```
void
swap(
```

for staff use only

–

"GCC hates me."

22. (2 points.) Suppose that Doug sees the warning below when he tries to compile his code.

```
implicit declaration of function 'printf'
```

Explain what the problem must be and how Doug can fix it.

23. (2 points.) Suppose that Michelle sees the error below when she tries to compile her code.

```
undefined reference to `GetInt'
```

Explain what the problem must be and how Michelle can fix it.

24. (1 point.) Suppose that Rose sees the message below when she tries to run `foo`.

```
Segmentation fault (core dumped)
```

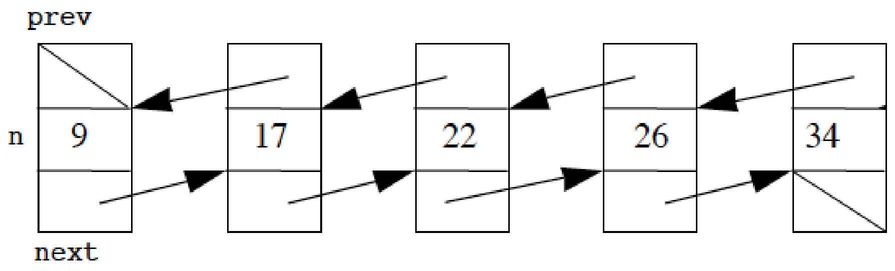
Explain what the problem might be.

for staff use only

–

Two Lists.

25. (3 points.) Consider the below depiction of a doubly linked list.



Suppose that each of the five nodes in this list is of type `node` and each of the numbers is of type `int`. Complete the below declaration of `node` in a manner consistent with this depiction.

```
typedef struct node
{




}
node;
```

26. (4 points.) For each algorithm below, specify an upper (*O*) and lower (Ω) bound on its running time. Assume that the linked lists and arrays in question are all of length *n*.

Algorithm	*O*	Ω
inserting into a **sorted** singly linked list		
searching a **sorted** singly linked list		
sorting a **sorted** array with Selection Sort		
sorting an **unsorted** array with Merge Sort		

for staff use only
–

C meets PHP.

27. (6 points.) PHP, a language we'll look at later this term, has a function called `ucwords` that capitalizes the first letter of each word in a string. That's just too exciting. Let's implement it now in C.

    Complete the implementation of `ucwords` below. Rather than modify `s` itself, `ucwords` should instead return a copy of `s` with the first letter of each word in that copy capitalized (if not already); all other characters, no matter their case, should remain unchanged.[4] For simplicity, assume that between each pair of adjacent words in `s` will be a single space (`' '`), that `s` will not begin or end with spaces, and that all characters in `s` will be alphabetical (or spaces). For instance, we might pass your function `"homer j simpson"` or `"homer J simpson"` (both of which should become `"Homer J Simpson"`) or even `"hOmEr j sImPsOn"` (which should become `"HOmEr J SImPsOn"`), but we won't pass it `"homer j. simpson"` (because of the period). Do not assume that `s` will contain only two or three words; it may contain zero or more. And `s` may very well be `NULL`. (D'oh!) You are welcome to call any function you know to exist in C; no need to `#include` any header files.

```
char *
ucwords(const char *s)
{
```

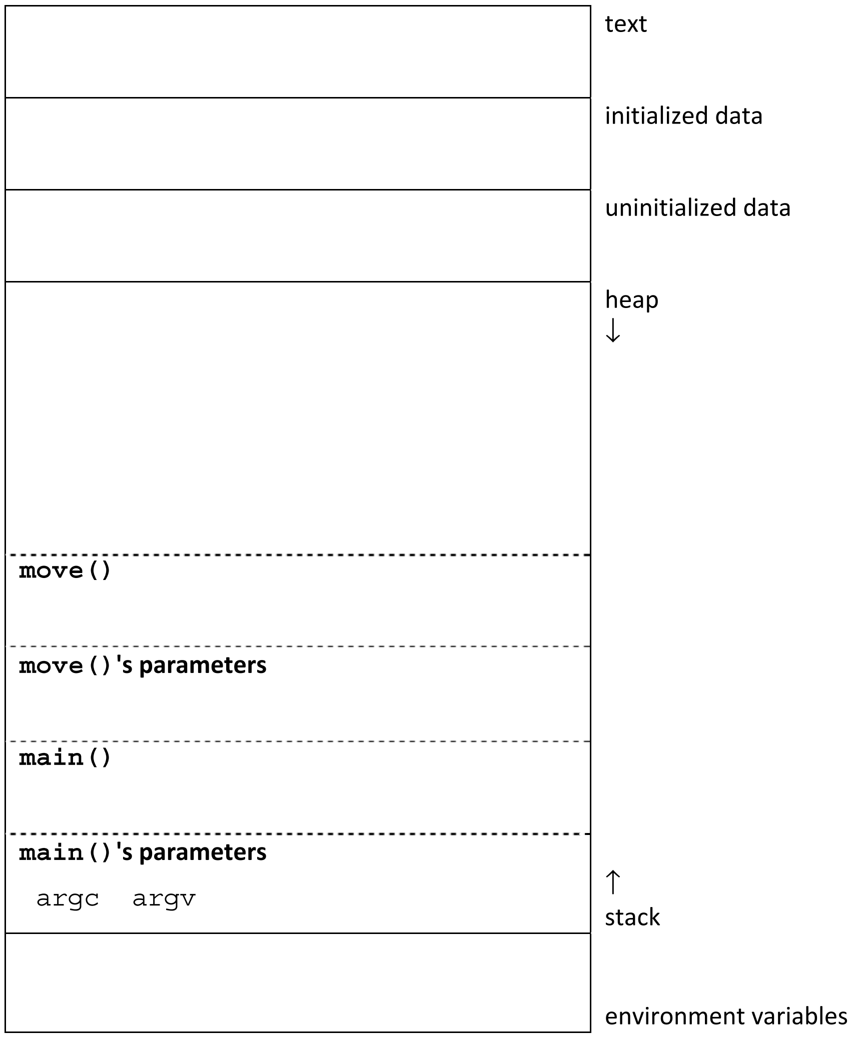
[4] We've declared `s` as `const` (*i.e.*, constant) to emphasize that you should not modify `s` itself. But that doesn't mean you can't make a copy of it!

for staff use only

–

Distant Memory.

28. (4 points.) Included at the end of this quiz is `fifteen.c` from Problem Set 3's distro. Suppose that we paused execution of `fifteen` within `move()` (as via a breakpoint using GDB). Without worrying about specific addresses, tell us where each of `board` (line 33), `d` (line 34), `tile` (line 89), and `tile` (line 176) can be found in memory, generally speaking, by jotting down each of those symbols in the appropriate space below. For example, we've noted where `argc` and `argv` can be found. For symbols that belong in the same general area, do not worry about order (*e.g.*, we could have written `argc` to the right of `argv`). Because line 89's and line 176's symbols are identically named, make clear somehow which one goes where.



for staff use only

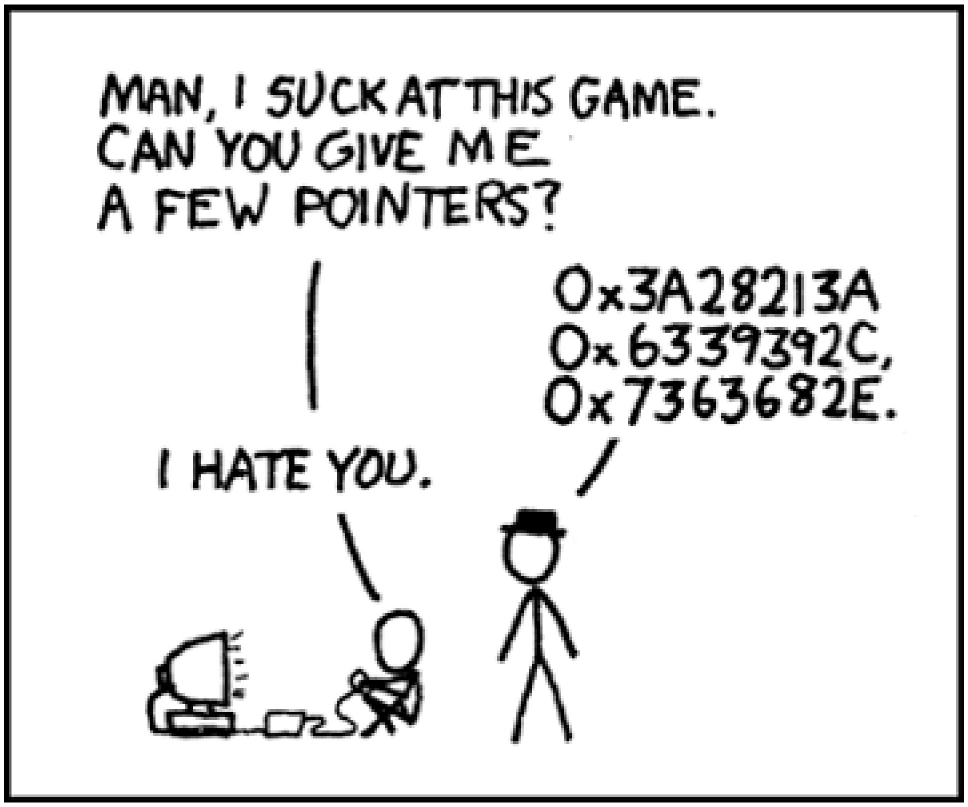
–

Extra Noncredit.

29. (0 points.) What does the program below print when executed?

```
main() { char *s="main() { char *s=%c%s%c; printf(s,34,s,34); }"; printf(s,34,s,34); }
```

kthxbai



`http://xkcd.com/138/`

Scrap Paper.

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

```c
/***************************************************************************
 * fifteen.c
 *
 * Computer Science 50
 * Problem Set 3
 *
 * Implements The Game of Fifteen (generalized to d x d).
 *
 * Usage: fifteen d
 *
 * whereby the board’s dimensions are to be d x d,
 * where d must be in [DIM_MIN,DIM_MAX]
 *
 * Note that usleep is obsolete, but it offers more granularity than
 * sleep and is simpler to use than nanosleep; ‘man usleep‘ for more.
 ***************************************************************************/

#define _XOPEN_SOURCE 500

#include <cs50.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>


// constants
#define DIM_MIN 3
#define DIM_MAX 9


// global board
int board[DIM_MAX][DIM_MAX];
int d;


// prototypes
void clear();
void greet();
void init();
void draw();
bool move();
bool won();


int
main(int argc, char *argv[])
{
    // greet user with instructions
    greet();

    // ensure proper usage
    if (argc != 2)
    {
        printf("Usage: %s d\n", argv[0]);
        return 1;
    }

    // ensure valid dimensions
    d = atoi(argv[1]);
    if (d < DIM_MIN || d > DIM_MAX)
    {
        printf("Board must be between %d x %d and %d x %d, inclusive.\n",
               DIM_MIN, DIM_MIN, DIM_MAX, DIM_MAX);
        return 2;
    }

    // initialize the board
    init();

    // accept moves until game is won
    while (true)
    {
        // clear the screen
        clear();

        // draw the current state of the board
        draw();

        // check for win
        if (won())
        {
            printf("ftw!\n");
            break;
        }

        // prompt for move
        printf("Tile to move: ");
        int tile = GetInt();

        // move if possible, else report illegality
        if (!move(tile))
        {
            printf("\nIllegal move.\n");
            usleep(500000);
        }

        // sleep thread for animation’s sake
        usleep(500000);
    }

    // that’s all folks
    return 0;
}


/*
 * void
 * clear()
 *
 * Clears screen using ANSI escape sequences.
 */

void
clear()
{
    printf("\033[2J");
    printf("\033[%d;%dH", 0, 0);
}


/*
 * void
 * greet()
 *
 * Greets player.
 */

void
greet()
{
    clear();
    printf("WELCOME TO THE GAME OF FIFTEEN\n");
    usleep(2000000);
}


/*
 * void
 * init()
 *
 * Initializes the game’s board with tiles numbered 1 through d*d - 1
 * (i.e., fills 2D array with values but does not actually print them).
 */

void
init()
{
    // TODO
}


/*
 * void
 * draw()
 *
 * Prints the board in its current state.
 */

void
draw()
{
    // TODO
}


/*
 * bool
 * move(int tile)
 *
 * If tile borders empty space, moves tile and returns true, else
 * returns false.
 */

bool
move(int tile)
{
    // TODO
    return false;
}


/*
 * bool
 * won()
 *
 * Returns true if game is won (i.e., board is in winning configuration),
 * else false.
 */

bool
won()
{
    // TODO
    return false;
}
```