Name: *(as it would appear on official course roster)*	
Umail address: @umail.ucsb.edu	**section**
Optional: name you wish to be called if different from name above.	
Optional: name of "homework buddy" (leaving this blank signifies "I worked alone"	

1

h14

CS16 W17

h14: Chapter 14: Recursion

ready?	assigned	due	points
true	Tue 03/07 03:30PM	Tue 03/14 04:30PM	38

You may collaborate on this homework with AT MOST one person, an optional "homework buddy".

MAY ONLY BE TURNED IN IN THE LECTURE/LAB LISTED ABOVE AS THE DUE DATE, OR IF APPLICABLE, SUBMITTED ON GRADESCOPE. There is NO MAKEUP for missed assignments; in place of that, we drop the three lowest scores (if you have zeros, those are the three lowest scores.)

Read Chapter 14 and the lecture notes. If you do not have a copy of the textbook yet, there is one on reserve at the library under “COMP000-STAFF - Permanent Reserve”.

Please:

- **No Staples**.
- **No Paperclips**.
- **No folded down corners**.

PLEASE MARK YOUR HOMEWORK CLEARLY, REGARDLESS OF IF YOU WRITE IT OUT IN INK OR PENCIL! FOR BEST RESULTS, SAVE THIS PAGE AS A PDF, THEN PRINT THE PDF.

1.(2 pts) How does a recursive function know when to stop recursing?

When it encounters the base case, a recursion function knows to stop.

2.(3 pts) What is a LIFO scheme and how does it relate to stacks?

LIFO is an abbreviation of “Last In, First Out”, which means the most recently added element will be removed firstly. The way how stacks store and output elements is exactly “LIFO”.

3.(3 pts) What is stack overflow?

It is a situation where the call stack pointer exceeds the stack bound since the stack always has limited space. Infinite or very deep recursions can cause this.

2

h14

CS16 W17

4.(15 pts) Copy the binary search function found at this URL: https://github.com/ucsb-cs16-wi17/hw14

It has a few mistakes: fix them and get the program to work correctly. I recommend you do this by examining the code, compiling it, see what errors come up, then edit the code, and repeat the process until you have it working. Do NOT turn in the completed, fixed program. Instead write out in bullet points in the space below, EVERYTHING you had to fix to get the program working.

line 8 - Insert: #include <algorithm>   line 20 - Insert: sort(a, a + ARRAY_SIZE);

line20: const int final_index = ARRAY_SIZE - 1;

line27: search(a, 0, final_index, key, found, location);

line29: if (found)

line40: int mid = (first + last) / 2;

line42: found = true;

line48: search(a, mid + 1, last, key, found, location);

5.(15 pts) Write a recursive function program to find the *n*th element in the following arithmetic numerical sequence: **3, 11, 27, 59, 123, …**

Hint: You first have to figure out what is the recursive pattern. You also have to identify the base case.

```
int search(int n)
{
   if (n == 0) return 3;
   return search(n - 1) + pow(2, n+2);
}
```

```
int series (int n) {
    if (n==0) return 3;
    return  2*series(n-1)+5;
}
```