15-110: Principles of Computing, Spring 2018

Problem Set 5 (PS5)

Due: Friday, February 23 by 2:30PM via Gradescope Hand-in

HANDIN INSTRUCTIONS

Download a copy of this PDF file. You have two ways to fill in your answers:

1. **Just edit (preferred) - Use any PDF editor (e.g., Preview on Mac, iAnnotate on mobile, Acrobat Pro on pretty much anything) to typeset your answers in the given spaces. You can even draw pictures or take a picture of a drawing and import it in the correct place in the document. That's it. (Acrobat Pro is available on all cluster machines.)**

2. **Print and Scan - Alternatively, print this file, write your answers neatly by hand, and then scan it into a PDF file. This is labor-intensive and must be done by the deadline.**

Once you have prepared your submission, submit it on Gradescope. A link to Gradescope is provided in our Canvas course portal.

Fill in your answers ONLY in the spaces provided. Any answers entered outside of the spaces provided may not be graded. Do not add additional pages. We will only score answers in the given answer spaces provided. If we cannot read your answer or it contains ambiguous information, you will not receive credit for that answer.

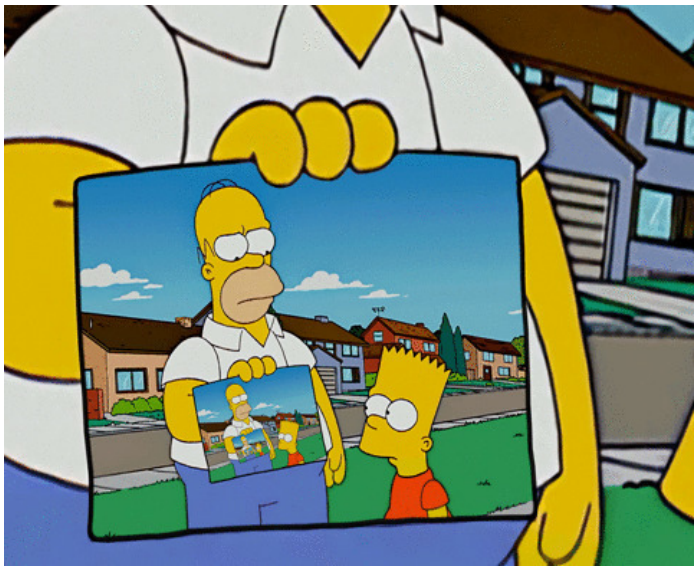
Be sure to enter your full name below along with your section letter (A, B, C, etc.) and your Andrew ID. Submit your work on Gradescope by 2:30PM on the Friday given above.

REMINDER: Sharing your answers with another student who is completing the assignment, even in another semester, is a violation of the academic integrity policies of this course. Please keep these answers to yourself.

Name (First Last) _____

Section _____ Andrew ID _____

1. (1 pt) Homer Simpson is holding up a picture to Bart as shown below. Briefly explain what big computational principle he is illustrating.



2. (2 pts) Let's reconsider the Towers of Hanoi problem. Recall that the priests had 64 discs, and once they solved the problem, the world would end. Thankfully, they didn't solve the problem, and we'll see why here.

   (a) The Towers of Hanoi problem requires 127 moves for 7 discs. Thinking recursively, how many moves are required for 8 discs? Briefly state how you used recursion to come up with your answer.

   (b) How many moves are required to move n discs as a function of n?

   (c) Now back to the priests. If they had 64 discs and moved one disc per hour (remember these were pretty heavy for priests), how long would it take for them to solve the problem if they didn't stop to do anything else? Express your answer in years.

3. (2.5 pts) Recall the recursive binary search algorithm discussed in class, with its Python implementation below.

```
def bs_helper(datalist, key, lower, upper):
    if lower + 1 == upper:    # base case: range empty
        return None
    mid = (lower + upper)//2
    if key == datalist[mid]:  # base case: found key
        return mid
    if key < datalist[mid]:
        return bs_helper(datalist, key, lower, mid)
    else:
        return bs_helper(datalist, key, mid, upper)

def bsearch(datalist, key):
    return bs_helper(datalist, key, -1, len(datalist))
```

Let datalist =
`[5, 12, 14, 19, 23, 27, 33, 39, 45, 56, 61, 70, 79, 81, 98]`

(a) Show the sequence of recursive calls that are made **and** the final returned result when we search for the key 12. The first two function calls are given for you.

```
bsearch(datalist, 12)

→ bs_helper(datalist, 12, -1, 15)

→
```

(b) Show the sequence of recursive calls that are made **and** the final returned result when we search for the key 50. The first two function calls are given for you.

```
bsearch(datalist, 50)

→ bs_helper(datalist, 50, -1, 15)

→
```

4. (1.5 pts) Consider the list [97, 58, 86, 14, 40, 71, 85, 39] which we want to sort using Merge Sort:

Merge Sort Algorithm:
1. Sort the first half of the list.
2. Sort the second half of the list.
3. Merge the two sorted halves to get a sorted list.

After steps 1 and 2, we get two sorted halves: a = [14, 58, 86, 97] and b = [39, 40, 71, 85].

(a) Show the merge process of step 3 to create a merged list c using the sorted lists above by tracing the merge algorithm. Circle the two values that are compared for each iteration and show which value gets appended to list c. The first line is shown for you.
Once you run out of data from one list, append the rest of the other list to c. You may not need all of the rows given.

a	b	c
14 58 86 97	39 40 71 85	**14**
14 58 86 97	39 40 71 85	
14 58 86 97	39 40 71 85	
14 58 86 97	39 40 71 85	
14 58 86 97	39 40 71 85	
14 58 86 97	39 40 71 85	
14 58 86 97	39 40 71 85	
14 58 86 97	39 40 71 85	
14 58 86 97	39 40 71 85	
14 58 86 97	39 40 71 85	

(b) How do you make the Merge Sort algorithm above recursive?

5. (2 pts) We wish to count how many multiples of 6 are in a list of integers. Here is a recursive algorithm we can follow:

    1. If the list is empty, there are no multiples of 6 in the list.
    2. Otherwise, if the first integer in the list is a multiple of 6, then the total amount of multiples of 6 in the list is 1 plus the number of multiples of 6 in the rest of the list.
    3. Otherwise, the total amount of multiples of 6 in the list is just the total amount of multiples of 6 in the rest of the list.

    By "rest of the list", we mean the list without the first element.

Complete the recursive implementation of this function in Python3:

```
def count_multiples_of_6(numlist):

    if ____________________________________:

        return 0

    elif ____________________________________________:

        return ____________________________________________

    else

        return ____________________________________________
```

6. (1 pt) The `zip` command has a `-r` option for recursive zip. For example, you could type:

```
zip –r lab.zip lab/*
```

Thinking recursively, how does this option work when you zip the lab directory?