

15-110: Principles of Computing, Spring 2018

Problem Set 6 (PS6)

Due: Friday, March 2 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. (1.5 pts) Consider the following two-dimensional table of positive integers.

4	3
12	5
26	11
5	9

- (a) Show how this table would be represented in Python as a two-dimensional list.

```
matrix = _____
```

- (b) Assuming that the parameter `matrix` always represents a two-dimensional list, complete the function below that computes and returns the number of odd integers in the matrix.

```
def count_odd(matrix):  
    count = 0  
    for row in range(0, len(matrix)):  
        for col in range(0, len(matrix[row])):  
            if _____:  
                count = _____  
    return count
```

- (c) For the matrix shown at the top of the problem, what are the values of:

`len(matrix)`

`len(matrix[0])`

2. (2 pts) At commencement, an organizer is determining how to seat graduates before they come up on stage for their diplomas. Students are called up to the stage alphabetically.


Algorithm 1: Students line up and are seated alphabetically in seats, leaving no seat open.

Algorithm 2: Students are seated in the order they arrive, leaving no seat open. Each student holds a card with the seat number of the student that follows them in alphabetical order, except the last student alphabetically. The organizer only keeps track of the first student alphabetically.

- (a) If a new graduating student arrives at commencement after everyone has been seated and he/she is first alphabetically, which algorithm makes seating that student faster? What needs to be done in each case?



- (b) If the organizer needs to call up the last student alphabetically for a special award, which algorithm allows the organizer to find this student more quickly? What needs to be done in each case?



3. (1.5 pt) An RPN expression can be stored in a list as follows:

```
rpn = [ 6, 2, "+", 9, 4, "-", "*", 1, 8, 2, "/", "+", "/" ]
```

Trace how the stack-based algorithm for RPN expressions computes the value of the given RPN expression stored as a list. (**Note:** We'll use integer division for "/".)

Complete a new stack trace whenever a number is pushed or popped to show how the stack progresses throughout the computation. The first three stack traces are shown for you. Use Courier 12 pt font to make the spacing easier to handle if you type answers.

		2												
6	6	8												
<hr/>														
Final result returned: _____														

4. (1.5 pts) A stack is a data structure with a LIFO (Last In First Out) property. That is, the last element you put in is the first one you can take out.

Recall that for stacks, the insert operation is called *push*. If your stack *s* is implemented using a list and you want to put *x* on the stack, *push* is done this way:

```
s.append(x)
```

This means that the end of the list is acting as the "top of the stack".

The remove operation is called *pop*. If your stack is implemented using a list, *pop* is done this way:

```
y = s.pop()
```

(The *pop* function without a parameter removes and returns the last element of the list since the end of the list is the "top of the stack". This is different than the *remove* function, which removes the element specified as its parameter but returns *None*.)

Now consider a *queue*, a data structure with a FIFO (First In First Out) property. In this case, the first element you put in is the first one you can take out. A queue is like a line you would stand in when waiting for service at a store checkout, when buying tickets at a movie theater, or when paying a vehicle toll at a bridge. With a queue, you *enqueue* (enter the queue) on to the rear of the queue, and you *dequeue* (depart the queue) from the front of the queue.

Suppose we represent a queue using a list named q such that the first element in the list (at index 0) is the front of the queue and the last element in the list (at index $len(q)-1$) is the rear of the queue.

- (a) Show how to enqueue an element stored in the variable z on to the rear of the queue q using Python, assuming that q is a list that behaves like a queue.

```
def enqueue(q, z):
```

```
    _____
```

- (b) Show how to dequeue an element from the front of the queue q into variable z and return this element using Python, assuming that q is a list that behaves like a queue. (CAREFUL: There is an extra condition needed to make sure this operation does not fail.)

```
def dequeue(q):
```

```
    if _____:
```

```
        return None
```

```
    else:
```

```
        z = _____ # get "front" element
```

```
        _____ # remove it from queue
```

```
    return z
```

5. (1.5 pts) A hash table has 11 buckets (i.e. its table size is 11). When strings are stored in the hash table, we use the following hash function to return the bucket that is used to store the string in:

```
def h(string_data, table_size):
    k = 0
    for i in range(0, len(string_data)):
        k = k * 256 + ord(string_data[i])
    bucket_number = k % table_size
    return bucket_number
```

In the function above, `ord(string_data[i])` returns the ASCII code of the i^{th} character of `string_data`. Here are the ASCII codes for the lowercase letters:

a	b	c	d	e	f	g	h	i	j	k	l	m
97	98	99	100	101	102	103	104	105	106	107	108	109
n	o	p	q	r	s	t	u	v	w	x	y	z
110	111	112	113	114	115	116	117	118	119	120	121	122

- (a) Given the hash function above, in which bucket would the following words be stored? Show the formula used to compute the bucket number. Do not write the final answer only.

The formula for the first element is shown for you. Also indicate if this string would collide with any string before it.

```
"can"
k = ((0 * 256 + 99) * 256 + 97) * 256 + 110 = 6513006
bucket_number = k % 11 = 5
```

```
"bag"
k = _____

bucket_number = k % 11 = _____ Collision?(y/n)_____
```

```
"bin"
k = _____

bucket_number = k % 11 = _____ Collision?(y/n)_____
```

```
"jug"
k = _____

bucket_number = k % 11 = _____ Collision?(y/n)_____
```

6. (2 pts) Based on Chapter 3 of *Blown To Bits*, answer the following questions about data.

- a. Identify the redacted text from the following sentence. (The reading gives you a clue on how to capture the text.)

Alan told Ada that his password for the computing server is [REDACTED] .

- b. A standard compact disc of music can hold up to 80 minutes of music. If the music were stored in a mp3 format instead of its current format, would the disc store more music or less music? Why?

- c. Find the secret person in the following company announcement. (HINT: Use the principle of **steganography** as explained in the text.)

The operator makes critical operations rapidly to improve network authentication.

- d. Herman puts a compact disc of data by itself in a sealed time capsule that is to be opened 100 years from now so it can be utilized. Let's assume that the CD does not decay or change in any way so it comes out exactly as it was put in. Based on the reading in the chapter, what is the main problem with storing the data for 100 years this way so others can use it later?