# ECS 32A: Practice Problem Set #1 for Exam #2

Instructor: Aaron Kaloti

Summer Session #1 2020

## Contents



## 1 Changelog

You should always refer to the latest version of the syllabus.

- v.1: Initial version.
- v.2: Fixed typo in problem #9.

## 2 Problems

Not all of the questions on here are of the exact types that I would ask on the real exam. For instance, problem #1 would never appear on an exam in this class, since everyone would just run the four lines in the Mu editor or Python IDLE or whatever.

### 2.1 Problem #1

What is the output of this program?

```
a = "aabbccddee"
print(a[6:])
print(a[2:5])
print(a[1:8:3])
```

### 2.2 Problem #2

What is the output of this program?

```
sentence = "What is up?"
print("x" in sentence)
print(sentence[2] * 3)
print(sentence[0] + "b")
```

## 2.3 Problem #3

Write down a string that, when passed as an argument to `foo()`, would cause the program to crash. If there is no such string, then say that.

```
def foo(a):
    return a[0]
```

## 2.4 Problem #4

Write a function called `bar` that takes as argument a string and returns `True` if the length of the string is greater than 5 and `False` otherwise.

Here are some examples of how your function should behave:

```
>>> bar("abcd")
False
>>> bar("abcde")
False
>>> bar("abcdef")
True
```

## 2.5 Problem #5

Remove as many lines of code as possible from the below program, without changing the output of this program. That is, whatever output that results from the user entering a specific input should be the same regardless of any changes that you make to the code shown below.

Write the letters of the lines you would cross off. You may only delete lines; you may not change or add lines. If multiple lines have the same letter, then that means you must either delete all of those lines or none at all. If you delete line B, then assume that the body of the `for` loop will be dedented.

If a given line has no associated letter, then that means I am not allowing you to cross it off.

```
def foo(string,target):
    i = 0                               # A
    for i in range(len(string)):        # B
        if i == len(string):            # C
            break                       # C
        if i == target:                 # D
            return string[i]            # D
        else:                           # E
            continue                    # E
        return True                     # F
    return False                        # G
```

## 2.6 Problem #6

Write a program that prompts the user to enter a string and a character. For the second input, if the user did not enter a character (i.e. if they entered nothing or entered a string larger than one character), then the program should inform them. Otherwise, the program should tell them if the given character is in the provided string. You are not allowed to use a loop.

## 2.7 Problem #7

Rewrite the below program such that it uses a `while` loop instead of a `for` loop.

```
for i in range(8,15,3):
    print(i)
```

## 2.8 Problem #8

Rewrite the below program such that it uses a `for` loop instead of a `while` loop.

```
i = 50
while 1000 > i:
    print(i)
    i += 1
```

## 2.9 Problem #9

Write a function that takes a string as argument and returns the number of characters in the string that are equivalent to the first character of that string.

## 2.10 Problem #10

Write a function called ‘foo‘ that takes as argument a string and returns ‘True‘ if the number of capital A’s in the string is greater than the number of capital B’s. Otherwise, the function should return ‘False.‘

*Hint*: This problem is doable with one loop; you don’t need to use two or more loops, nested loops, etc. That said, if you do come up with something needlessly complex, you still will get full credit if your solution is correct.

**You may not use any string method, except for ‘format‘. Note that ‘len()‘ is not a method, so ‘len()‘ is *allowed*. You may not import any module. You may not use any built-in functions that we have not discussed during class.**

Here are some examples of how your function should behave.

```
>>> foo("BAH")
False
>>> foo("BAHA")
True
>>> foo("BAHAB")
False
>>> foo("baha")
False
>>> foo("AAAA")
True
```

UCDAVIS
COMPUTER SCIENCE