CSCI 150 – Fall 2021

Tracing with the Heap

In preparation for the exam on Monday November 8, here are some practice Heap Tracing Examples. You can think of this as the first part of the Practice Exam for Exam #3. You are not required to do these or turn them in. However, if you turn them in by class time on Wednesday November 3, your instructor will grade them (see below), and this will replace any single missing/partially complete homework assignment.

We will go over these briefly as part of the Exam review on Friday November 5. The actual exam will contain about 3 tracing problems of similar levels of difficulty.

Grading Specifications:
You will earn a **Complete** provide that:

- all five problems are attempted,
- at least three a completely correct, showing all changes in the stack and heap as appropriate,
- a fourth has minor computational/careless error, but shows correct interactions with the heap, and
- on all five, there is no confusion about print vs return values.

You will earn a **Partially Complete** provided that:

- at least four are attempted,
- on at least three, interactions with the heap are correct, except for minor computational errors, and
- there is no confusion on at least two about print vs return values

You are **STRONGLY** encouraged to copy the code shown into a .py file or Kaggle notebook and run them. See if you printed output matches. Add additional print statements along the way to see if you are updating the heap/stack variables correctly. If you are confused, seek help from classmates, the CSCI tutors, or your instructor.

For all, you might find it useful to include

```
from typing import *
```

as the first line of your .py file or first cell in your Kaggle notebook

1.

```
def main1():
    a_list = [1, 2, 3]
    b_list = a_list
    temp_list = []
    for item in a_list:
        temp_list.append(item * 2)

    b_list.append(47)
    a_list[1] = -7

    print(a_list)
    print(b_list)
    print(temp_list)

main1()
```

2.

```
def main2():
    a_dict = {1: 'cat', 2: 'dog', 34: 'fish'}
    b_dict = a_dict
    temp_dict = {}
    for item in a_dict:
        temp_dict[item] = a_dict[item] + '!!'

    b_dict[100] = 'pig'
    a_dict[2] = 'snail'

    print(a_dict)
    print(b_dict)
    print(temp_dict)

main2()
```

3.

```
def main3():
    a_str = 'bye'
    b_str = a_str
    temp_str = ''
    for item in a_str:
        temp_str += item + '!'

    b_str += 'z'


    print(a_str)
    print(b_str)
    print(temp_str)

main3()
```

4.

```
def f1(a: Dict[str, int]) -> int:
    sum1 = 0
    for key in a:
        if a[key] >= 0:
            sum1 += a[key]
        else:
            a[key] = 0

    return sum1

def main4():
    b = {'Seme' : 23, 'Ferrer' : 12, 'Wilson' : -7}
    print(f1(b))
    print(b)

main4()
```

5.

```
def g1(s: str) -> int:
    if 'a' in s:
        s = 'boo'

    return len(s)

def g2(lst: List[int]):
    s = 'exam'

    if len(lst) < len(s):
        print('Too short')
    else:
        i = 0
        for char in s:
            lst[i] = g1(char)
            i += 1


def main5():
    a_list = [6,2,9,8]
    g2(a_list)
    print(a_list)

main5()
```