MTH 3300
E. Fink

3300 Problems, Section 4: Nested Loops, `while` Loops

1. What will print out when the following code runs?

```
x = 0
for i in range(1,3):
   for j in range(1,3):
      x = x + j
   print(i)
print(x)
```

2. What will print out when the following code runs?

```
x = 1
for i in range(3):
   for j in range(x):
      print(j+x, end = "")
   x += 1
   print(j)
print(x)
```

3. What will be displayed when the following code runs?

```
x = 0
for r in range(4):
   for c in range(r):
      print("X", end = "")
   print("")
```

4. What will the following code display?

```
import turtle
t = turtle.Turtle()
scr = turtle.Screen()
for i in range(10):
    t.forward(10)
    t.left(90)
    t.forward(10)
    t.right(90)
scr.mainloop()
```

Recall that the turtle starts out facing to the right!

5. What will the following code display?

```
import turtle
t = turtle.Turtle()
scr = turtle.Screen()
for i in range(4):
    t.forward(10)
    for j in range(2):
        t.right(90)
        t.forward(10)
    t.left(90)
scr.mainloop()
```

Recall that the turtle starts out facing to the right!

6. Write code which allows the user to input a list of names, until they type in `"XXX"` (which should *not* be part of the list). At this point, the list should be printed out backwards.

7. Write a program that will accept a stream of integers from the user, until the user inputs the same integer twice. At this point, the program should print out `REPEAT!!`.

8. Complete the following code so that it allows the user to input an ODD positive integer n, and then prints out the following "T" shape (shown here with $n = 5$):

```
*****
  *
  *
  *
  *
```

The shape should contain n stars in the top row, and n stars arranged vertically, with the middle star in the top row being the same as the top star in the vertical arrangment.

FOR FULL CREDIT: do not use string multiplication or `.format()`.

```
n = int(input("How many rows: "))
```

Complete the following code so that it allows the user to input a positive integer n and a positive integer k, and then prints out the following figure (shown here with $n = 5$ and $k = 2$):

```
++---
++---
++---
++---
++---
```

9. Write code that asks the user for a value of n, and then prints n rows of the following "hairpin" shape (shown here for $n = 6$):

```
*
**
* *
*  *
*   *
*    *
```

For full credit, DO NOT USE string multiplication or `.format()`; instead, just print single spaces and single stars.

The shape should contain n rows, and each row should display n characters. Each row should have k `+`'s, with the remaining entries in each row being `-`'s.

FOR FULL CREDIT: do not use string multiplication or `.format()`.

```
n = int(input("How many rows: "))
k = int(input("How many +'s per row: "))
```

10. *Without using string multiplication*, create code that creates a 11×11 square that looks like

```
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
 * * * * *
* * * * * *
```

11. I write code to print out the values that appear twice in the following array:

```
myList = [4,6,7,6]
```

I use the following code, expecting `6` to appear on the console.

```
x = len(myList)
for i in range(0,x):
   for j in range(i,x):
      if myList[j] == myList[i]:
        print(myList[i])
```

It doesn't work like I planned it. What prints out *instead*? (Do your walkthrough carefully!) How could I fix this code, so that it works for this list (and any other list that contains duplicates, but no numbers that appear more than twice)?

12. Suppose that you are provided with two lists, named `arr1` and `arr2` – for example,

```
arr1 = [2,8,4,6,12,10]
arr2 = [3,6,9,18,15,12]
```

Complete this code to create a new list which only contains the elements present in both lists. In this example, the new list produced should be `[6, 12]`. The code should work even if `arr1` and `arr2` are replaced with different lists; you may assume that neither list contains duplicates.

13. Determine what the code below displays:

```
i = 1
j = 20
while i < j:
   i += 2
   j -= 1
print(j)
```

14. Explain in words what interesting phenomenon happens when the following loop is run (and why), in one sentence.

```
x = ["a", "b", "c"]
index = 0
while index <= 2:
   print(x[index])
```

15. What will print out from the following loop?

```
x = 2
y = 3
while y < 10:
   x += 1
   if x == y:
      print(x)
   y += 3
   print(y)
```

16. What will print out from the following loop?

```
i = 1
j = 2
while i < 20:
   i += j
   if j == 4:
      break
print(i)
```

17. The following loop contains no *syntax* errors, but there is still an issue with the way that it is coded. Briefly explain what will happen when you run this code:

```
i = 1
x = 0
while i != 10:
   x += i
   i += 2
print(x)
```

18. Convert the following code to code that utilizes only a `while` loop:

```
my_list = [123,456,789,1011]

for num in my_list:
    if num > 500:
        print(my_list * 2)
```

19. Write code which evaluates 100 terms of the sum $1^2 + 11^2 + 21^2 + 31^2 + \ldots$, using ONLY a `while` loop (no `for` loops!).

20. a. Write code which computes the sum $1 \cdot 2 + 4 \cdot 5 + 7 \cdot 8 + \ldots + 100 \cdot 101$, using a `for` loop.

    b. Compute the same sum as in part a using only a `while` loop (no `for` loops!).

21. a. Write code which computes the sum $2 + \sin(4) + 6 + \sin(8) + 10 + \sin(12) + \ldots + 98 + \sin(100)$, using a `for` loop.

    b. Compute the same sum as in part a using only a `while` loop (no `for` loops!).

22. The following loop uses a flag to keep asking the user to input an integer, until they input an integer that is **between 1 and 100**. What lines should fill in the blank so that the flag works properly? *Assume that the user always inputs an integer.*

```
success = False

while not success:
    entry = int(input("Enter an integer (but if it's not between 1 and 100, I'll just ask again): "))
    -------------
    -------------

print(entry)
```

23. Write code which asks the user to input an initial value for a variable `x`. The program should repeatedly take `x`, print it, and double it. It should do this until `x` is greater than 1000, at which point it should print out the number of times it doubled.

    For example, if the user entered `100`, it should print

```
100
200
400
800
It doubled 3 times.
```

    (It stops at 800 because 1600 would be greater than 1000.)

24. I start with the following code:

```
x = int(input("Enter a number: "))
y = int(input("Enter a number: "))
```

    Write additional code which will repeatedly print the values of `x` and `y`, and then "move" `x` and `y` by subtracting 2 from `x` and adding 5 to `y`. It should do this until `x` is less than `y`; when `x` is finally less than `y`, it should print out these values of `x` and `y`, and it should print out the number of moves it took for `x` to become less than `y`.

    For example, if the user entered `20` and `1`, the program should print

```
20 1
18 6
16 11
14 16
It took 3 moves.
```

    (It stops because 14 is less than 16.)

25. Write a program that asks the user to enter 10 *positive* integers, and then print out the *largest one that is LESS THAN 100* to the console. Use a LOOP for full credit. You may assume that all numbers are positive, and at least one number is less than 100.

    Example run (where the numbers in the first row are user input):

```
Enter 10 integers:  30 5 140 98 100 10 55 120 40 96
The largest less than 100 is:  98
```

26. Write a program that allows the user to input a succession of integers until the *same number comes in three in a row.* At that point, the program should stop and print `Threepeat!`

    So, a sample run of the program may look like this:

```
Number?  5
Number?  19
Number?  5
Number?  5
Number?  7
Number?  7
Number?  7
Threepeat!
```

where here, all the numbers are user input, and all the rest is produced by the program.