`Directions:` The exam is 120 minutes long. Please read each question carefully.

When asked to write code, you should write working Python code that has correct syntax. You should explain in 1-2 sentences what the idea for your solution is or write next to your code what it is doing. This will increase your chances of getting full/partial credit.
Use the backs of the pages if needed.

Last Name: ______________________

First Name: ______________________

Student ID #: ______________________

Question	Points	Score
1	20	
2	20	
3	20	
4	20	
5	20	
6	20	
7	20	
8	20	
9	0	
Total:	160	

1. (20 points) Write down the output of the following programs.

1.
```
x = 1
s = 0
for i in range(8):
    s += x
    x *= 10
print(s)
```

2.
```
def f(n):
    if n > 0:
        return n + g(n - 1)
    return 1

def g(n):
    return f(n) - n

print(f(11))
```

3.
```
xs = [0,2,7]
x = sum(map(lambda x: 2 ** x, xs))
print(x)
```

4.
```
# fin
def g(n):
    if n == 0:
        return []
    return [n % 2] + g(n // 2)

print(g(133))
```

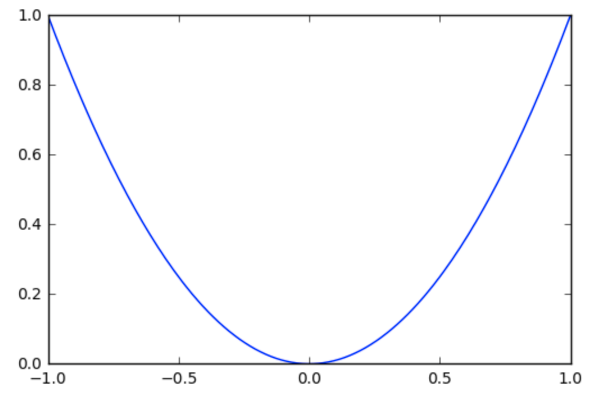
2. (20 points) Produce the following lists without using for or while loops.

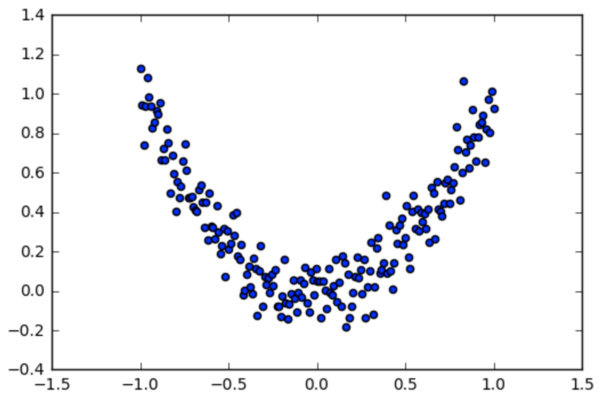
   1. `[11, 101, 1001, 10001, 100001, 1000001, 10000001, 100000001, 1000000001]`

   2. `[0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9]`

   3. `[0, 101, 2, 103, 4, 105, 6, 107, 8, 109, 10, 111, 12, 113, 14]`

3. (20 points) Write code that will produce the following graphs (or something that looks like it; use `plt.plot(X, Y)` and `plt.scatter(X, Y)`).

   1. 

   2. 

4. (20 points) Complete the code below to implement the function `square(n)` that will return a numpy array with the numbers $1, 2, \ldots, n^2$ arranged in a square.

   Examples:

```
In:   square(3)
Out:  array([[1, 2, 3],
             [4, 5, 6],
             [7, 8, 9]])


In:   square(4)
Out:  array([[ 1,  2,  3,  4],
             [ 5,  6,  7,  8],
             [ 9, 10, 11, 12],
             [13, 14, 15, 16]])
```

```
def square(n):
    X =
    return
```

   Complete the code below to implement the function `square_2(n)`, which produces the same square as the above, but with all entries except the ones at the edges set to zero.

```
In:   square_2(n)
Out:  array([[ 0,  1,  2,  3,  4,  5,  6],
             [ 7,  0,  0,  0,  0,  0, 13],
             [14,  0,  0,  0,  0,  0, 20],
             [21,  0,  0,  0,  0,  0, 27],
             [28,  0,  0,  0,  0,  0, 34],
             [35,  0,  0,  0,  0,  0, 41],
             [42, 43, 44, 45, 46, 47, 48]])
```

```
def square_2(n):
    X = squa(n)


    return X
```

5. (20 points) Implement a function `isprime(n)` that will return `True` if an integer `n` is prime and `False` otherwise.

6. (20 points) Recall the `Polynomial` class from the homework that stores a polynomial as a list of its coefficients. Implement the `__eval__(self, x)` function that returns a new polynomial which represents the sum of the polynomials self and other.

```
class Polynomial():
    def __init__(self, xs):
        self.coeffs = xs

    def __repr__(self):
        if self.coeffs == []:
            return "0"
        c = ""
        for i, x in enumerate(self.coeffs):
            c += str(x) + "x" + "^" + str(i) + "␣+␣"
        return c[:-3]

    def eval(self, x):
```

7. (20 points) Write a function `is_rearranged_palindrome(n)` that takes an integer `n` and returns `True` if and only if the digits of `n` can be rearranged to make a palindrome. (You can assume that `n` has an even number of digits if it makes things easier)

8. (20 points) Your buddy Joker has given you a coin that you think might be unfair. You can flip the coin by calling the function `flip()` which returns either 0 or 1 (for heads or tails). You can call `flip()` at most 1000 times. Write code that will compute your best estimate for the probability of `flip()` being tails.

9. (extra-credit, 20 points) Only near-perfect answers will receive the extra-credit. Please don't attempt this problem if you haven't already finished all other problems and checked your answers.

   Monty Hall Problem: You are in a game-show. The host tells you that you can choose between the three doors in front of you. Only one of the doors has a pot of gold behind it. You choose door 1. The host opens one of the other doors and shows you that it did not have the pot of gold. The host then gives you the choice to stick to your original choice of door 1 or swich your choice to the remaining other door. Should you switch?

   Write a simulation that evaluates the success rate of two strategies: (a) always switching to the remaining other door, (b) always sticking with door 1.