1. Trace the output of the following code:

```python
def add_3(num):
    return num + 3

def times_2(num):
    return num * 2

def subtract_4(num):
    return num - 4

def add_nums(num1, num2):
    return num1 + num2

def convert_to_str(num):
    return str(num)

def concat_str(str1, str2):
    return str1 + str2

num1 = 5
num2 = 8
new_num1 = times_2(add_3(num1))
new_num2 = subtract_4(times_2(num2))

print(add_nums(num1, num2))
print(add_nums(new_num1, new_num2))
print(concat_str(num1, num2)))
print(concat_str(convert_to_str(new_num1), \
      convert_to_str(new_num2)))
```

2. Write a function `get_valid_passwords(passwords)` that accepts `passwords`, a list of potential password strings, as a parameter and returns a list of valid password strings. For a password to be valid, it must follow these rules:
    a. It must be between 8 and 20 characters long.
    b. It must contain at least one uppercase letter (A-Z).
    c. It must contain at least one digit (0-9).
    d. It must contain at least one special character (!@#$%^&*()_+).

   To solve this problem, you may want to use the following methods:
    - `str.isupper()`: returns True if all characters in the string are upper case, False otherwise
    - `str.digit()`: returns True if all characters in the string are digits, False otherwise

   Hint: Write a helper function to check for each rule!

   For example, the following call to the function
   ```
   passwords = ["Password123!", "weak", "Str0ngP@ssw0rd",
                "12345678", "Qwerty@123"]
   get_valid_passwords(passwords)
   ```

   would return
   ```
   ['Password123!', 'Str0ngP@ssw0rd', 'Qwerty@123']
   ```

3. Write the function `rotate_list(num_list, shift_n)` that accepts two parameters, a list `num_list` and an integer `shift_n`, and creates a new list where each element is shifted to the right by `shift_n` indices (including wraparound). The function should then return this new list.

   For example:
   ```
   rotate_list([1,2,3,4], 1) returns [4,1,2,3]
   rotate_list([4,3,2,6,5], 2) returns [6,5,4,3,2]
   rotate_list([1,2,3], 0) returns [1,2,3]
   rotate_list([1, 2, 3], -1) returns [2,3,1]
   ```

4. Write the function `histogram(scores)` that accepts a parameter `scores`, a non-empty list of integers between 0 and 100, inclusive, representing exam scores, and prints a string representing a histogram of that data. In this histogram, we will have buckets for scores 0-9, 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90++. To print out our histogram, we will print only the non-empty buckets in order. For each nonempty bucket, we will print a star indicating a count for that bucket..

   For example, the following call to the function:
   ```
   scores = [73, 62, 91, 74, 100, 77]
   histogram(scores)
   ```
   Will print:
   ```
   60-69: *
   70-79: ***
   90++ : **
   ```