Question 1. [5 marks]

For each code fragment in the table below, select the answer that best describes the printed output, or the error that occurs when the code is run.

```
days = ['Mon', 'Tues']
days.append('Wed') + ['Thurs', 'Fri']
print(days)
```

(A) `['Mon', 'Tues']`
(B) `['Mon', 'Tues', 'Wed']`
(C) `['Mon', 'Tues', 'Thurs', 'Fri']`
(D) `['Mon', 'Tues', 'Wed', 'Thurs', 'Fri']`
(E) some other list is printed
(F) an error occurs because + is not defined for lists
(G) an error occurs because + is not defined for non-list and list

---

```
s = 'Valentine\'s' + 'Day'
print(s[8:13])
```

(A) `esDay`
(B) `es Da`
(C) `es Day`
(D) `e'sDa`
(E) `e's D`
(F) `e's Da`
(G) some other characters are printed
(H) an error occurs during the assignment statement

```
L = ['roses', 'red', 'violets',
     'blue', 'sugar', 'sweet']
s = ''
for i in range(len(L) // 3):
    s = s + L[i * 3][1]
print(s)
```

(A) `rb`
(B) `ol`
(C) `rrv`
(D) `oei`
(E) `rrvbss`
(F) `oeiluw`
(G) some other characters are printed
(H) an error occurs inside the loop body

```
print(int(2.999) > int('2'))
```

(A) `True`
(B) `False`
(C) an error occurs converting one of the arguments to int

```
print('CAT!'.find('!') == 3 or 7 / 0 > 0)
```

(A) `True`
(B) `False`
(C) a `ZeroDivisionError` occurs
(D) another error occurs

```
1. G    2. D    3. B    4. B    5. A
```

Question 2. [4 marks]

Part (a) [2 marks]

You need to write a function that returns whether there is an odd digit in a string at a particular index. Both the string and the index are parameters to the function. Do Step 1 of the Function Design Recipe: write two example function calls and their expected results. As always, choose a good name for your function.

You do not need to write any other steps of the Function Design Recipe.

```
    """
    >>> has_odd_digit('54', 1)
    False
    >>> has_odd_digit('1233', 3)
    True
    """
```

Part (b) [2 marks]

You've decided you want to write a function that returns whether a string contains only characters that appear in another string. Below are some example calls such as you might produce during Step 1 of the Function Design Recipe. Fill in the function header, including the type contract. As always, select good parameter names.

You do not need to write a description. You do not need to write the function body.

```
def contains_letters(s1: str, s2: str) -> bool

    """
    >>> contains_chars('cat', 'ewoks are cuter')
    True
    >>> contains_chars('abracadabra', 'abcdr')
    True
    >>> contains_chars('a', 'bcd')
    False
    """
    # DO NOT WRITE THE FUNCTION BODY
```

Question 3. [4 marks]

Complete the following function according to its docstring.

```
def latte_order(shots: int, milk: str, flavour: str) -> str:
    """Return the latte order with the given espresso shots, milk, and flavour,
    formatted as follows:
    <shots> shot <flavour> latte with <milk> milk

    There should exactly one space between each word in the order, and no
    extra leading or trailing spaces.

    If milk is 'A', replace it with 'almond', and if milk is 'S', replace it
    with 'soy'.

    Precondition: milk is one of 'A', 'S', '2%', '1%'

    >>> latte_order(2, 'A', 'vanilla')
    '2 shot vanilla latte with almond milk'
    >>> latte_order(1, '2%', 'pumpkin spice')
    '1 shot pumpkin spice latte with 2% milk'
    >>> latte_order(3, 'S', '')
    '3 shot latte with soy milk'
    """
    s = str(shots) + ' shot '
    if flavour != '':
        s = s + flavour + ' '

    s = s + 'latte with '

    if milk == 'A':
        s = s + 'almond'
    elif milk == 'S':
        s = s + 'soy'
    else:
        s = s + milk

    s = s + ' milk'
    return s
```

Question 4. [3 marks]

Fill in the box with the while loop condition required for the function to work as described in its docstring.

```
def find_lowercase(word: str) -> int:
    """Return the index of the first lowercase character in word, or the length of word
    if it does not contain any lowercase characters.

    >>> find_lowercase('cats')
    0
    >>> find_lowercase('123Hello')
    4
    >>> find_lowercase('CAT!')
    4
    """
    i = 0
    while [                                                        ]:
        i = i + 1
    return i
```

```
    i = 0
    while i < len(word) and not word[i].islower():
        i = i + 1
    return i
```

Question 5. [5 marks]

Complete the function body below according to its docstring. Hint: consider using `range` on your answer.

```
def has_3_consecutive_digits(s: str) -> bool:
    """Return True if and only if s has 3 digits that appear consecutively
    (i.e., next to each other) in s, and False otherwise.

    >>> has_3_consecutive_digits('abc123')
    True
    >>> has_3_consecutive_digits('a1b2c3')
    False
    """

    for i in range(len(s) - 2):
        if s[i:i+3].isdigit():
            return True
    return False
```

Question 6. [3 marks]

Complete the following function according to its docstring, **without using method** `str.count`.

```
def get_percent(s: str, ch: str) -> float:
    """Return the percentage of characters in s that are equal to ch.
    The percentage should be between 0.0 and 1.0.

    Precondition: len(s) >= 1 and len(ch) == 1

    >>> get_percent('hello', 'l')
    0.4
    >>> get_percent('Testing 123', 'z')
    0.0
    >>> get_percent('Valentines', 'a')
    0.1
    """

    num_ch = 0
    for char in s:
        if ch == char:
            num_ch = num_ch + 1

    return num_ch / len(s)
```