Remember Version 3 Reflection Activity

Q1 Modify the given code such that it satisfies the requirement of Limiting Literals as specified in Section 5 of the Software Quality Test Document.

Given Code

```
# generate and display our list of words for the game
words = random.sample(all_words, 4)
answer = random.choice(words)
answer_start_letter = answer[0]
for word in words:
    os.system(clear_command)
    print('-'*80)
    print(' Remember The Word')
    print('-'*80)
    print(word)
    time.sleep(2)
```

Modified Code

```
number_of_words = 4

words = random.sample(all_words, number_of_words)
answer = random.choice(words)
answer_start_letter = answer[0]

header_border = "*" * 80
header_content = "Remember The Word"

pause_time = 2

for word in words:
  os.system(clear_command)
  print(header_border)
  print(header_content)
  print(header_border)
  print(word)
  time.sleep(pause_time)
```

Q2 Modify the given code such that it satisfies the requirement of Replacing all adjacent line groups with iteration as specified in Section 4 of the Software Quality Test Document.

Given Code

```
# display instructions
print('A sequence of words will be displayed.')
print('You will be asked which word starts with a particular letter.')
print('You win if you enter the right word.')
print('You have 3 attempts to guess the word correctly.')
```

Modified Code

```
instructions = ["A sequence of words will be displayed.",
                "You will be asked which word starts with a particular letter.",
                "You win if you enter the right word.",
                 "You have 3 attempts to guess the word correctly."]

for phrase in instructions:
  print(phrase)
```

Q3 Identify the token kind (choose from : operator, delimiter, str literal, int literal, float literal, identifier, keyword) for EACH token that is bolded and underlined in each statement of the given code segment.

Given Program Segment	Type of Token
`words = random.sample(all_words, 4)`	int literal
`correct_answer = random.choice(words)`	delimiter
`start_letter = correct_answer[0]`	int literal
`pause_time = 2`	delimiter
`for word in word_list:`	keyword
`    os.system(clear_command)`	identifier
`    print(word)`	identifier
`    time.sleep(pause_time)`	identifier

Q4 Use the Python built-in len function to compute the index of the last element in word_list. Use the computed index to modify the given code such that the variable correct_answer is bound to the last element of the list instead of being randomly chosen.

Given Code

```
word_list = random.sample(all_words, 4)
correct_answer = random.choice(word_list)
```

Modified Code

```
word_list = random.sample(all_words, 4)
correct_answer = word_list[len(word_list) - 1]
```

Q5 Edit this block of code so that the program sleeps for half a second for each letter in the word being displayed. That way, very long words are displayed for more time than short words.

Given Code

```
for word in words:
    os.system(clear_command)
    print(header_border)
    print(header_content)
    print(header_border)
    print(word)
    time.sleep(2)
```

Modified Code

```
for word in words:
  os.system(clear_command)
  print(header_border)
  print(header_content)
  print(header_border)
  print(word)
  time.sleep(0.5 * len(word))
```

Q6 For each of the following Python statements, write the type of object the underlined and bolded identifier is bound to (select from the types: function, module, int, str, list, tuple, bool, etc.). In the case of function type, indicate whether it is a normal function or a method.

Python Statement	Type of Object
`print(word)`	str
`time.sleep(pause_time)`	module
`os.system(clear_command)`	function
`word_list = random.sample(all_words, 4)`	list
`answer = random.choice(word_list)`	str
`answer_start_letter = answer[0]`	str
`answer_start_letter = answer[0]`	int
`time.sleep(pause_time)`	function
`result = guess == answer`	bool

Q7 Modify this code so that the correct answer is the longest word in word_list, rather than a random word, using a for loop.

Given Code

```
word_list = random.sample(all_words, 4)
correct_answer = random.choice(word_list)
```

Modified Code

```
word_list = random.sample(all_words, 4)

longest_word = word_list[0]

for word in word_list:
    if len(word) > len(longest_word):
        longest_word = word

correct_answer = longest_word
```

```
word_list = random.sample(all_words, 4)

longest_word = word_list[0]
index = 1

while index < len(word_list):
    if len(word_list[index] > len(longest_word):
        longest_word = word_list[index]
    index = index + 1

correct_answer = longest_word
```

```
word_list = random.sample(all_words, 4)

longest_word = word_list[0]

for index in range(1, len(word_list)):
    if word_list[index] > longest_word:
        longest_ word = word_list[index]

correct_answer = longest_word
```

Q8 Modify this code so that the player is repeatedly prompted to guess the correct answer until they actually get it correct, using a while loop.

Given Code

```
guess = input('What word begins with the letter '+answer_start_letter+'? ')

# display feedback
if guess == answer:
   print('Congratulations, you are correct.')
```

Modified Code

```
correct = False

while correct = False:
   guess = input("What word begins with thhe letter " + answer_start_letter + "?")
   correct = guess == answer

print("Congratulations you are correct.")
```

Q9 Complete the missing code segment so that it will calculate the length of the longest line of text in a .txt file, using a for loop.

Given Code

```
max_length = 0
filename = 'random_file.txt'
file = open(filename, 'r')


...

print('The longest line was ' + str(max_length) + ' characters long.')
```

Modified Code

```
max_length = 0
filename = "random_file.txt"
filemode = "r"
file = open(filename, filemode)
content = file.read()
file.close()

list_lines = content.splitlines()
longest_line = list_lines[0]

for line in list_lines:
  if len(line) > len(longest_line)
    longest_line = line

max_length = len(longest_line)

print("The longest line was " +
str(max_length) + " characters long.")
```