Q-07

Due No due date **Points** 18 **Questions** 18 **Time Limit** 30 Minutes **Allowed Attempts** Unlimited

Take the Quiz Again

Attempt History

	Attempt	Time	Score
KEPT	Attempt 6	29 minutes	14 out of 18
LATEST	Attempt 6	29 minutes	14 out of 18
	Attempt 5	30 minutes	13 out of 18
	Attempt 4	30 minutes	11.75 out of 18
	Attempt 3	19 minutes	13 out of 18
	Attempt 2	30 minutes	11.5 out of 18
	Attempt 1	30 minutes	5.25 out of 18

Correct answers are hidden.

Submitted Jun 29 at 1:32am

Question 1

1 / 1 pts

The highlighted section below illustrates.
If the variable str has any characters

```
Given: the variable str is a string (may be empty)
Create the counter variable, initialized to -1
If the variable str has any characters then
{
    Set counter to 0
    Create the variable current-character as a character
    Place the first character in str into current-character
    While more-characters and current-character not a period
    {
       Add one to (or increment) the counter variable
       Store the next character from str in current-character
    }
    If current-character is a period then
       Add one to the counter to account for the period.
    Else
      Set counter to -2
}
If counter is -1 the string was empty
Else if counter is -2 there was no period
```

- ○ a necessary condition
- ○ a boundary condition
- ● a loop guard
- ○ None of these
- ○ a postcondition
- ○ an intentional condition

Question 2

1 / 1 pts

Which line represents the ***intentional bounds*** in this loop?

```
1.      string s("Hello CS 150");
2.      while (s.size())
3.      {
4.         if (s.at(0) == 'C') break;
5.         s = s.substr(1);
6.      }
7.      cout << s << endl;
```

- ○ 5

[✓ 4]

None of these

2

Question 3

1 / 1 pts

Below is the illustration from the loop building strategy. The ***highlighted lines*** represent.
Store the next character from str in current-character:

```
Given: the variable str is a string (may be empty)
Create the counter variable, initialized to -1
If the variable str has any characters then
{
    Set counter to 0
    Create the variable current-character as a character
    Place the first character in str into current-character
    While more-characters and current-character not a period
    {
       Add one to (or increment) the counter variable
       Store the next character from str in current-character
    }
    If current-character is a period then
       Add one to the counter to account for the period.
    Else
      Set counter to -2
}
If counter is -1 the string was empty
Else if counter is -2 there was no period
```

[✓ advancing the loop]

goal precondition

goal operation

bounds precondition

loop bounds

loop postcondition

Question 4

1 / 1 pts

Which line ***advances the loop***?

```
1.     string s("Hello CS 150");
2.     while (s.size())
3.     {
4.        if (s.at(0) == 'C') break;
5.        s = s.substr(1);
6.     }
7.     cout << s << endl;
```

[✓ 5]

2

None of these

4

Question 5

1 / 1 pts

Below is the illustration from the loop building strategy. The ***highlighted lines*** represent.
Create the variable current-character as a character:

```
Given: the variable str is a string (may be empty)
Create the counter variable, initialized to -1
If the variable str has any characters then
{
    Set counter to 0
    Create the variable current-character as a character
    Place the first character in str into current-character
    While more-characters and current-character not a period
    {
       Add one to (or increment) the counter variable
       Store the next character from str in current-character
    }
    If current-character is a period then
       Add one to the counter to account for the period.
    Else
      Set counter to -2
}
If counter is -1 the string was empty
Else if counter is -2 there was no period
```

- ○ loop bounds
- [✓ bounds precondition]
- ○ advancing the loop
- ○ loop postcondition
- ○ goal operation
- ○ goal precondition

Question 6

1 / 1 pts

The highlighted section below illustrates.
current-character not a period:

```
Given: the variable str is a string (may be empty)
Create the counter variable, initialized to -1
If the variable str has any characters then
{
    Set counter to 0
    Create the variable current-character as a character
    Place the first character in str into current-character
    While more-characters and current-character not a period
    {
       Add one to (or increment) the counter variable
       Store the next character from str in current-character
    }
    If current-character is a period then
       Add one to the counter to account for the period.
    Else
      Set counter to -2
}
If counter is -1 the string was empty
Else if counter is -2 there was no period
```

- ○ a loop guard
- [✓ an intentional condition]
- ○ a boundary condition
- ○ a necessary condition
- ○ None of these
- ○ a postcondition

Question 7

1 / 1 pts

The highlighted section below illustrates.
While more characters:

```
Given: the variable str is a string (may be empty)
Create the counter variable, initialized to -1
If the variable str has any characters then
{
    Set counter to 0
    Create the variable current-character as a character
    Place the first character in str into current-character
    While more-characters and current-character not a period
    {
       Add one to (or increment) the counter variable
       Store the next character from str in current-character
    }
    If current-character is a period then
       Add one to the counter to account for the period.
    Else
      Set counter to -2
}
If counter is -1 the string was empty
Else if counter is -2 there was no period
```

- ○ a postcondition
- ○ a loop guard
- ○ an intentional condition
- [✓ a necessary condition]
- ○ None of these
- ○ a boundary condition

Question 8

1 / 1 pts

Match each item with the correct question below.

What must I change in the test to go to the next iteration?	advance the loop
What must I do to enter the loop?	bounds precondition
Has my loop reached its goal?	loop postcondition
Can my loop be entered at all?	loop guards

Question 9

1 / 1 pts

Below is the illustration from the loop building strategy. The ***highlighted lines*** represent.
While more-characters and current-character not a period:

```
Given: the variable str is a string (may be empty)
Create the counter variable, initialized to -1
If the variable str has any characters then
{
    Set counter to 0
    Create the variable current-character as a character
    Place the first character in str into current-character
    While more-characters and current-character not a period
    {
       Add one to (or increment) the counter variable
       Store the next character from str in current-character
    }
    If current-character is a period then
       Add one to the counter to account for the period.
    Else
      Set counter to -2
}
If counter is -1 the string was empty
Else if counter is -2 there was no period
```

- ○ goal operation
- ○ loop postcondition

[✓ loop bounds]

goal precondition

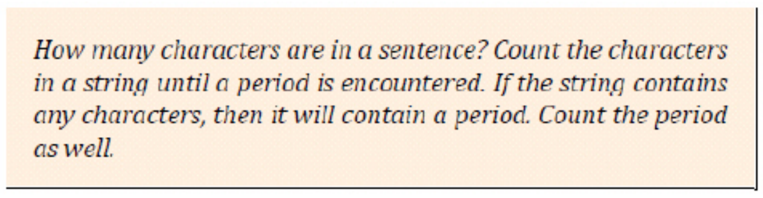
bounds precondition

advancing the loop

Question 10

1 / 1 pts

Look at the problem statement below. The __________ of the loop is to count the number of characters in a sentence.



How many characters are in a sentence? Count the characters in a string until a period is encountered. If the string contains any characters, then it will contain a period. Count the period as well.

bounds

None of these

plan

[✓ goal]

Question 11

1 / 1 pts

In H05, here is the pseudocode for the loop body. What code would turn an ASCII character into its digit value?

```
sum <- 0
number <- 0
for each character in str
   Set current character -> ch
   If ch is a digit then
      digit <- ascii-to-decimal(ch)
      number <- number * 10
      number <- number + digit
   Else
      sum <- sum + number
      number <- 0
```

`int digit = ch - "0";`

[✓ `int digit = ch - '0';`]

`int digit = ascii_to_decimal(ch);`

`int digit = static_cast<int>(ch);`

None of these answers is correct

Question 12

1 / 1 pts

In H05, here is the pseudocode for the loop body. What line of code needs to appear immediately after the loop body to make the algorithm complete?

```
sum <- 0
number <- 0
for each character in str
   Set current character -> ch
   If ch is a digit then
      digit <- ascii-to-decimal(ch)
      number <- number * 10
      number <- number + digit
   Else
      sum <- sum + number
      number <- 0
```

- ○ `number = number + sum;`
- [✓ `sum += number;`]
- ○ `number = number + digit;`
- ○ `sum = number + digit;`
- ○ None of these answers is correct

Incorrect

Question 13

0 / 1 pts

In `H05`, here is the pseudocode for the loop body. What code would you use to update the number, when a character was a digit?

```
sum <- 0
number <- 0
for each character in str
   Set current character -> ch
   If ch is a digit then
      digit <- ascii-to-decimal(ch)
      number <- number * 10
      number <- number + digit
   Else
      sum <- sum + number
      number <- 0
```

- ○ `number = number * 10 + digit;`
- ○ `number = number + digit * 10;`
- ○ `number = number * (10 + digit);`
- [✓ None of these answers is correct]
- ○ `number *= 10 + digit;`

Question 14

1 / 1 pts

In `H05`, here is the pseudocode for the loop body. What code would you write to "grab the current character"?

```
sum <- 0
number <- 0
for each character in str
   Set current character -> ch
   If ch is a digit then
      digit <- ascii-to-decimal(ch)
      number <- number * 10
      number <- number + digit
   Else
      sum <- sum + number
      number <- 0
```

- ○ `char ch = char[i];`
- ○ None of these answers is correct
- ○ `string ch = str.substr(i, 1);`
- [✓ `char ch = str.at(i)`]

`char ch; str.at(ch);`

Incorrect

Question 15

0 / 1 pts

The of `everyNth()` from `H06` is solved by using which pattern?

- Guarded Loop Pattern
- Growing a String Pattern
- Alternative Action Pattern
- Symmetric Loop Pattern
- [✓ Asymmetric Loop Pattern]

Question 16

1 / 1 pts

Here is an implementation of `prefixAgain()` from `H06`. What is its problem?

```
bool prefixAgain(const string& str, int n) {
    string prefix = str.substr(0, n);
    for (size_t i = 0, len = str.size(); i < len; ++i) {
        string word = str.substr(i, n);
        if (word == prefix) { return true; }
    }
    return false;
}
```

- There is no problem. It works correctly and is efficient.
- It does not compile because you can't use == with strings
- It doesn't compile because there is no `else` with the `if`
- It compiles and runs without crashing, but never produces the correct output
- [✓ It compiles and runs without crashing, but doesn't always produce the correct output]

Incorrect

Question 17

0 / 1 pts

Here is an implementation of `zipZap()` from `H06`. What is its problem?

```
string zipZap(const string& str) {
    string result;
    size_t len = str.size(), i = 0;
    if (len < 3) return str;
    while (i < len - 2) {
        string subs = str.substr(i, 3);
        if (subs.at(0) == 'z' && subs.at(2) == 'p') {
            result += "zp";
            i += 3;
        } else {
            result += subs.front();
            ++i;
        }
    }
    result += str.substr(i);
    return result;
}
```

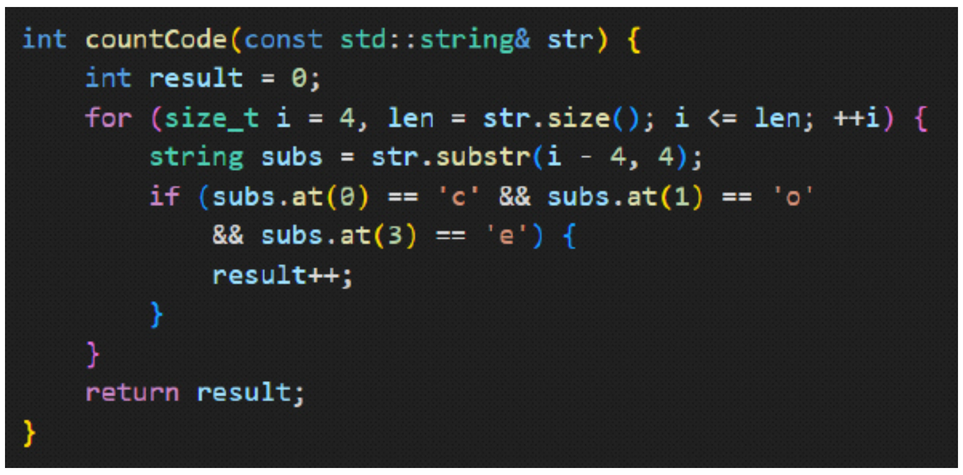
- There is no problem. It works correctly and is efficient.
- It does not compile
- [✓ It does not produce the correct output for strings of size less than 3]
- It produces the correct output, but is less efficient than it could be
- It should use `subs.at(0)` instead of `subs.front()` since it's more efficient

Incorrect

Question 18

0 / 1 pts

Here is an implementation of `countCode()` from `H06`. What is its problem?



```cpp
int countCode(const std::string& str) {
    int result = 0;
    for (size_t i = 4, len = str.size(); i <= len; ++i) {
        string subs = str.substr(i - 4, 4);
        if (subs.at(0) == 'c' && subs.at(1) == 'o'
            && subs.at(3) == 'e') {
            result++;
        }
    }
    return result;
}
```

- It does not produce the correct output
- There is no problem. It works correctly and is efficient.
- It compiles, but the loop condition should be `i < len;`
- It does not compile
- [✓ It produces the correct output but performs more iterations than required]