Q-10

Due No due date **Points** 10 **Questions** 8 **Time Limit** 30 Minutes **Allowed Attempts** Unlimited

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	19 minutes	9 out of 10

ⓘ Correct answers are hidden.

Submitted Jun 29 at 2:50am

Question 1

1 / 1 pts

What is the value of ***r("xxhixx")***?

```
string r(const string& s)
{
  if (s.empty()) return "";
  if (s.at(0) == 'x') return 'y' + r(s.substr(1));
  return s.at(0) + r(s.substr(1));
}
```

- Stack overflow
- xxyyxx
- xyxyhixyxy
- **yyhiyy**
- yxyxhixyyx

Incorrect

Question 2

0 / 1 pts

What does this function do?

```
int mystery(int n)
{
  if (n == 1) return 1;
  return n * mystery(n-1);
}
```

- Computes the Fibonacci number n
- Computes the Gauss series (sum) of 1..n
- Computes the Factorial number n
- Produces a stack overflow
- **Computes the reverse of the input n**

Question 3

1 / 1 pts

What is the value of ***r("hello")***?

```
string r(const string& s)
{
    if (s.size() > 1) {
```

```
        string t = s[0] == s[1] ? "" : "*";
        return s[0] + t + r(s.substr(1));
    }
    return s;
}
```

- (●) "h*e*ll*o"
- ( ) Stack overflow
- ( ) "hel*lo"
- ( ) "*h*e*ll*o"
- ( ) "h*e*ll*o*"

Question 4

1 / 1 pts

Which of the following is a key requirement to ensure that recursion is successful?

- ( ) A recursive function should not call itself except for the simplest inputs.
- ( ) There should be special cases to handle the most complex computations directly.
- ( ) A recursive solution should not be implemented to a problem that can be solved iteratively.
- (●) Every recursive call must simplify the computation in some way.

Question 5

1 / 1 pts

What is the value of ***r(3)***?

```
int r(int n)
{
   if (n < 2) { return 1; }
   return n * r(n - 1);
}
```

- ( ) 120
- ( ) 24
- (●) 6
- ( ) 2

Question 6

1.5 / 1.5 pts

Examine the code below and match the statements following it.

```
int mystery3(int n) {
    if (n < 2) return 1;
    return n * mystery3(n - 1);
}
```

The algorithm implemented is: Factorial

mystery3 completes for all inputs True

mystery3 is a recursive wrapper	False
mystery3 returns the correct answer for all inputs	False

mystery3 is an implementation of the Factorial algorithm. It completes for all inputs, but negative inputs produce the wrong output. It is efficient and it is not a wrapper. if (n < 2) is a base case.

Question 7

1.5 / 1.5 pts

Examine the code below and match the statements following it.

```
int mystery1(int n, int a, int b) {
    if (n == 0) return a;
    if (n == 1) return b;
    return mystery1(n - 1, b, a + b);
}

int mystery2(int n) {
    return mystery1(n, 0, 1);
}
```

mystery2 is a recursive wrapper	True
mystery2 completes for all possible inputs	False
if (n == 0) is a recursive case	False
These functions illustrate how inefficient recursion is.	False

mystery2 is a recursive wrapper around the recursive helper mystery1. Together they implement the Fibonacci sequence in an efficient manner. mystery2 will not complete for any negative inputs.

Question 8

2 / 2 pts

Examine the code below and match the statements following it.

```
int mystery3(int n) {
    if (n < 2) return 1;
    return n * mystery3(n - 1);
}
```

mystery3 has a stack overflow for some numbers.	False
mystery3 correctly implements its algorithm	True
if (n < 2) is a . . .	base case

mystery3 is efficient True

mystery3 is an implementation of the Factorial algorithm. It completes for all inputs, but negative inputs produce the wrong output. It is efficient and it is not a wrapper. if (n < 2) is a base case.