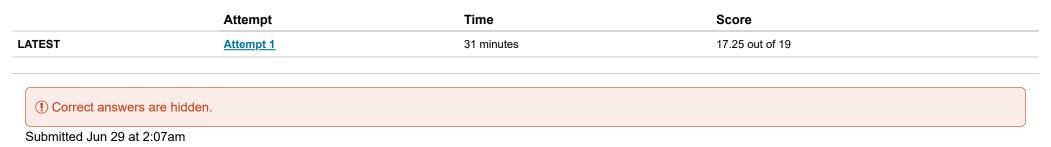
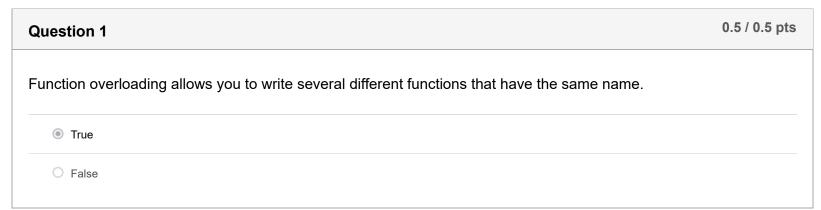
Due No due date Points 19 Questions 20 Time Limit 45 Minutes Allowed Attempts Unlimited

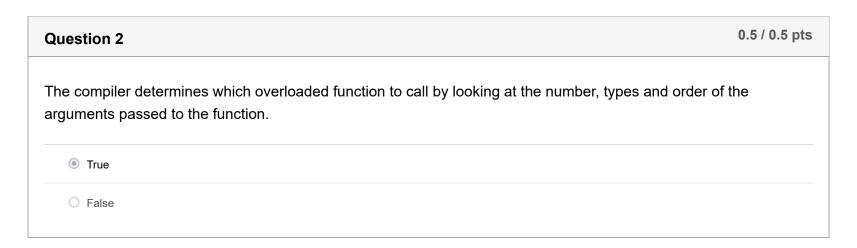
Take the Quiz Again

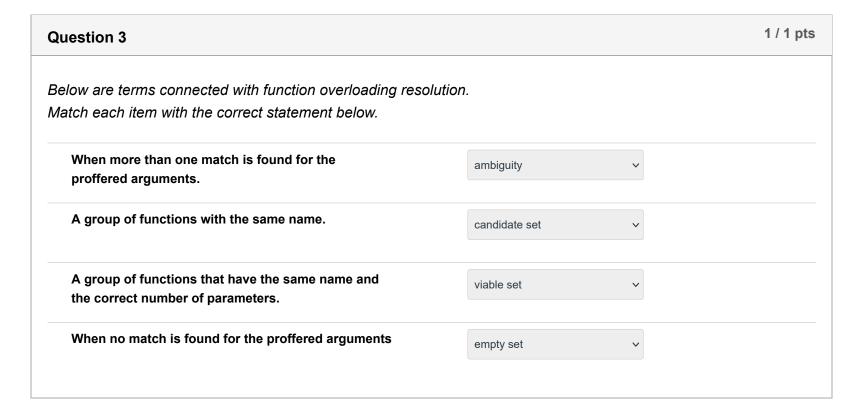
Attempt History

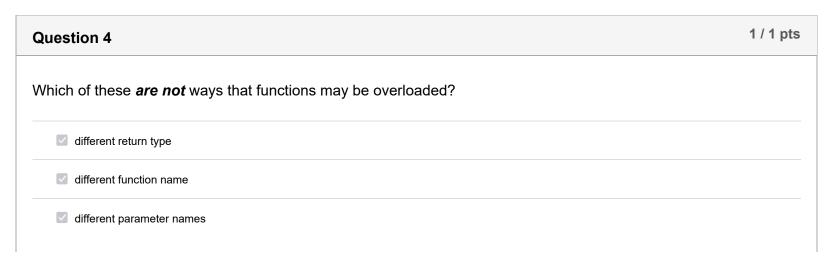












☐ different number of parameters		
☐ different parameter types		
different order of parameter types.		

Question 5	/ 1 pts
What prints?	
<pre>void fn(int, double, double&) { cout << "A" << endl; } void fn(int, int, double&) { cout << "B" << endl; } void fn(int, int, double) { cout << "C" << endl; } void fn(int, int, int) { cout << "D" << endl; }</pre>	
<pre>int main() { auto n = 3.5; fn(1, 2.5, n); }</pre>	
O Syntax error: no candidates	
O C	
O D	
O Syntax error: ambiguous	
ОВ	
A	

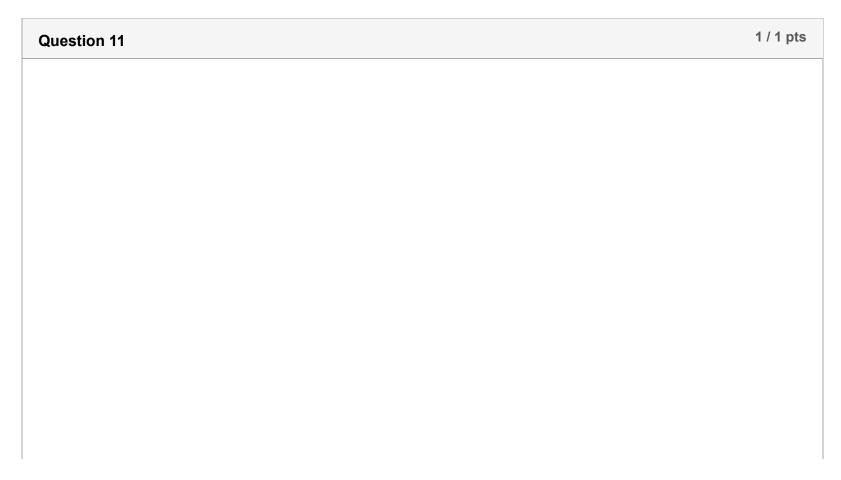
Question 6	1 / 1 pts
You can call a single function in several different ways by giving the function:	
default arguments	
O overloaded arguments	
O default parameters	
O reference parameters	
O optional parameters	

Question 7	0.5 / 0.5 pts
Default arguments let you call a single function in several different ways.	
● True	
○ False	

Question 8	0.5 / 0.5 pts
Default arguments appear only in the function implementation.	
O True	









To allow f() to change the argument passed here, the parameter str should be declared as:

void f(. . . str);int main()
{

string s = "hello"; f(s);}

It is not possible for f() to change the argument passed here.

const string

string&

const string&

string&

string&

Question 12	1 / 1 pts
What prints here?	
<pre>auto a = 'A';</pre>	
switch (a)	
{	
case 64: cout << "?";	
case 65: cout << "A";	
<pre>case 66: cout << "B"; }</pre>	
cout << endl;	
O ?	
O Prints nothing	
O Does not compile	
Ов	
O A	

Question 13	1 / 1 pts

```
What prints here?

auto a = 2;
switch (a)
{
    case 1: cout << "1"; break;
    case 2: cout << "2"; break;
    default: cout << "3";
}
cout << endl;

    1
    123
    0 2
    Does not compile
```

Question 14	1 / 1 pts
What prints here?	
<pre>auto a = 3, b = 3; cout << (a == b ? "panda": "tiger") << endl;</pre>	
O Crashes when run	
panda	
O Does not compile	
O Undefined behavior	
O tiger	

```
What prints here?

auto a = 3, b = 3;
cout << (a != b ? "panda": "tiger") << endl;

Does not compile

Crashes when run

Undefined behavior

itiger

panda
```

```
Question 16

Assume that the input is 4 4 3 2 5. What will print?

int i = 1;
int n;
cin >> n;
do
```

```
{
    i++;
    cin >> n;
}
while (n % 2);
cout << i << endl;

    3

    4

    Does not compile

    infinite loop

    2
```

Parti

Partial Question 17

0.25 / 1 pts

In H09, the fourth version of the overloaded read() functions could be implemented like this. What is true about this implementation?

```
bool read(char& c, char sentinel)
{
    cin.get(ch);
    if (c == sentinel || cin.fail()) return false;
    return true;
}
```

- ☐ The function will return false if cin runs out of input
- ☑ If the character read is not the sentinel character, it will return true
- $\hfill \square$ The function will skip any whitespace characters it encounters
- ☐ The function will read a single character if one exists
- ☐ If the character read is the sentinel, it will return false

Question 18 1 / 1 pts

In H09, the second version of the overloaded read() functions could be implemented like this. What happens with the call read("", x, true) when the user enters 23.57dog?

```
inline string hundreds(int n) { return digit(n, "CDM"); }
string digit(int digit, const string& symbols) { ... }
cout << hundreds(4) << endl;</pre>
```

- The function reads 23.57, removes the remaining input and returns false
- O The function returns false because 23.57dog is not a double
- O The function reads 23.57, leaves dog in input and returns false
- The function reads 23.57, removes the remaining input and returns true
- The function reads 23.57, leaves dog in input and returns true

Because the parameter n is a double, reading stops when it encounters the d in dog. Because the parameter ln is set to true, the remaining input on the line is removed.

Question 19 1 / 1 pts

In H09, the fourth version of the overloaded read() functions could be implemented like this. What is true about this implementation?

```
std::string barCode(int zip); { return ""; }
std::string codeForDigit(int digit); { return ""; }
int checkDigit(int zip); { return ""; }
```

- If the character read is the sentinel, it will return false
- If the character read is not the sentinel character, it will return true
- ☐ The function will fail if cin runs out of input
- ☐ The function will skip any whitespace characters it encounters
- ▼ The function will read a single character if one exists

If you are out of input, then the function will block, waiting for another character. However, when used with redirection, the function will fail when it reaches the end of input. This version will not skip any whitespace.

Incorrect Question 20 0 / 1 pts

In H09, the first version of the overloaded read() functions could be implemented like this. What happens with the call read("", x) when the user enters 23.57dog?

inline string tens(int n) { return digit(n, "XLC"); }
string digit(int digit, const string& symbols) { . . . }
cout << tens(7) << endl;</pre>

- The function reads 23, removes the remaining input and returns true
- O The function reads 23, leaves .57dog in input and returns true
- O The function reads 23.57, leaves dog in input and returns true
- The function reads 23.57, removes the remaining input and returns true
- O The function returns false because 23.57dog is not an int

Because the parameter n is an int, reading stops when it encounters the decimal point. Because the parameter ln is set to false, its default value, the remaining input on the line remains.

