Q-09

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

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	31 minutes	17.25 out of 19

Correct answers are hidden.

Submitted Jun 29 at 2:07am

Question 1

0.5 / 0.5 pts

Function overloading allows you to write several different functions that have the same name.

- (•) True
- ( ) False

Question 2

0.5 / 0.5 pts

The compiler determines which overloaded function to call by looking at the number, types and order of the arguments passed to the function.

- (•) True
- ( ) False

Question 3

1 / 1 pts

Below are terms connected with function overloading resolution.
Match each item with the correct statement below.

When more than one match is found for the proffered arguments.	ambiguity
A group of functions with the same name.	candidate set
A group of functions that have the same name and the correct number of parameters.	viable set
When no match is found for the proffered arguments	empty set

Question 4

1 / 1 pts

Which of these ***are not*** ways that functions may be overloaded?

- [x] different return type
- [x] different function name
- [x] different parameter names

- [ ] different number of parameters
- [ ] different parameter types
- [ ] different order of parameter types.

Question 5

1 / 1 pts

What prints?

```
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; }

int main()
{
    auto n = 3.5;
    fn(1, 2.5, n);
}
```

- [ ] Syntax error: no candidates
- [ ] C
- [ ] D
- [ ] Syntax error: ambiguous
- [ ] B
- [x] A

Question 6

1 / 1 pts

You can call a single function in several different ways by giving the function _____:

- [x] default arguments
- [ ] overloaded arguments
- [ ] default parameters
- [ ] reference parameters
- [ ] optional parameters

Question 7

0.5 / 0.5 pts

Default arguments let you call a single function in several different ways.

- [x] True
- [ ] False

Question 8

0.5 / 0.5 pts

Default arguments appear only in the function implementation.

- [ ] True

False

Question 9

2 / 2 pts

Examine the following variables and function calls
Match each item with the correct statement below.

```
int able = 3;
int baker = f1(able);
cout << able << baker << endl; // 64

string charlie;
f2("hello", charlie);
cout << charlie << endl; // Hello Carl
```

Returned value	baker
Output argument (parameter)	charlie
Input argument (parameter)	hello
Input/output argument (parameter)	able

Question 10

1 / 1 pts

Examine this code. Which is the best prototype?

```
int age;
string name = read("Enter your name, age: ", age);
```

- `string read(const string&, int)`
- None of these
- `string read(string, int);`
- `string read(const string, int&)`
- `string read(const string&, int&)`

Question 11

1 / 1 pts

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`

Question 12

1 / 1 pts

What prints here?

```
auto a = 'A';
switch (a)
{
    case 64: cout << "?";
    case 65: cout << "A";
    case 66: cout << "B";
}
cout << endl;
```

- ?
- Prints nothing
- Does not compile
- B
- **AB**
- 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;
```

- 3
- 1
- 123
- 2
- Does not compile

Question 14

1 / 1 pts

What prints here?

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

- Crashes when run
- panda
- Does not compile
- Undefined behavior
- tiger

Question 15

1 / 1 pts

What prints here?

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

- Does not compile
- Crashes when run
- Undefined behavior
- tiger
- panda

Question 16

1 / 1 pts

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

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
- [x] If the character read is not the sentinel character, it will return `true`
- [ ] 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;
```

- The function reads `23.57`, removes the remaining input and returns `false`
- The function returns `false` because `23.57dog` is not a `double`
- The function reads `23.57`, leaves dog in input and returns `false`
- [x] 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 ""; }
```

- [x] If the character read is the sentinel, it will return `false`
- [x] 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
- [x] 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;
```

- (•) The function reads `23`, removes the remaining input and returns `true`
- ( ) The function reads `23`, leaves `.57dog` in input and returns `true`
- ( ) The function reads `23.57`, leaves `dog` in input and returns `true`
- ( ) The function reads `23.57`, removes the remaining input and returns `true`
- ( ) 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.