Due no later than 9:00pm on Sunday 9/12.

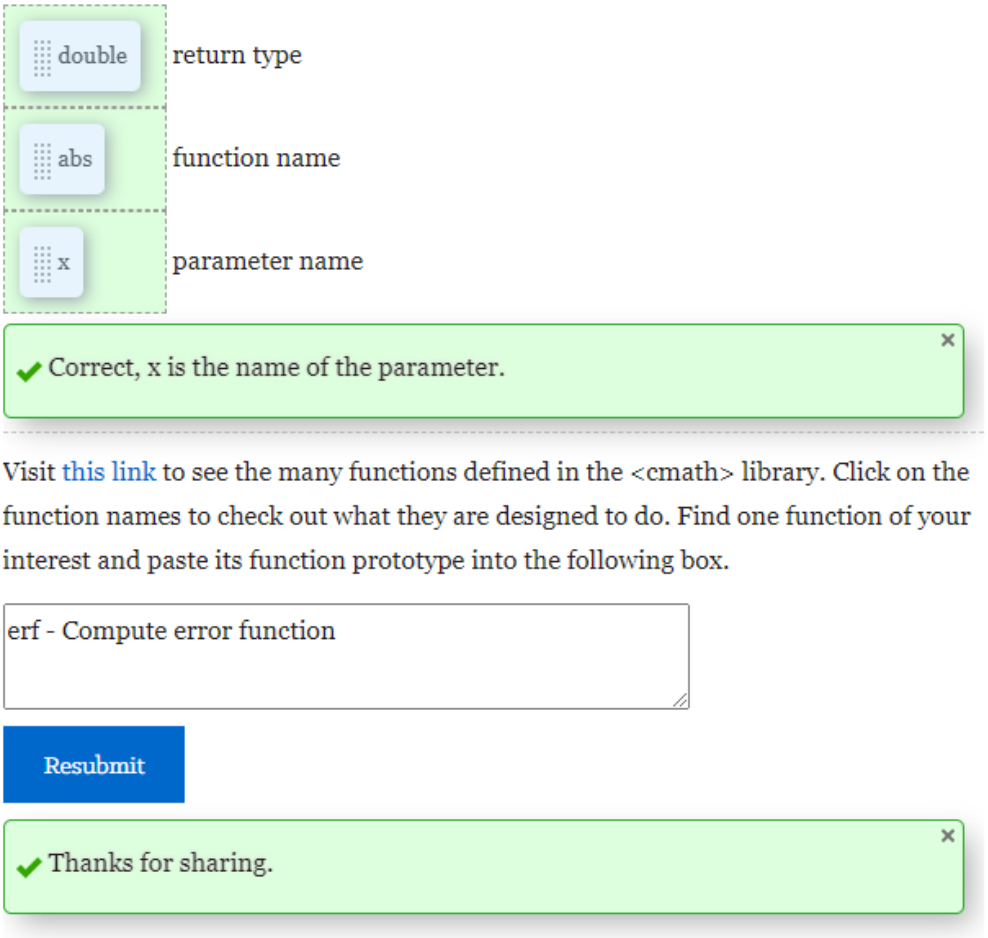
As you read the following OLI pages and complete the interactive activities, capture the screenshots of the completed activities and replace the respective screenshots in the document.

- Page 12 Math library functions
- Page 13 Program specific functions
- Page 15 The char data type
- Page 16 The string data type

When you are ready to submit the assignment, download the document in PDF and submit the PDF file on Cougar Courses as the proof for your work.

Page 12 Math library functions

LBD function prototype



LBD calling pow

When a function has two parameters, the compiler matches the arguments in a function call with the parameters based on position.

Match the numbers from the function call `pow(3.0, 2.0)` with their respective parameters.

3.0 base

2.0 exponent

✔ Correct, the second argument matches the second parameter.

What would the function call pow(3.0, 2.0) return?

- 6.0
- 9.0
- 8.0

✔ Correct; pow(3.0, 2.0) returns $3.0^{2.0}$.

LBD calling more <cmath> functions

Mark all of the function calls that will return 5.0.

- [ ] round(4.3)
- [ ] round(5.8)
- [x] round(4.6)
- [x] round(5.4)

Check My Answer

✔ Correct! `round(5.4)` and `round(4.6)` returns 5.0

```
double sqrt (double x);
```

Image Credit

Given the above function prototype, mark all valid function calls below.

- [ ] sqrt(2.5, 3.2)
- [x] sqrt(9.5 - 2.1)
- [x] sqrt(9.99)

Check My Answer

✔ Correct, the argument for the `sqrt` function may be a value or an expression that evaluates to a double value. It does not work with two arguments.

According to the documentation, what would be returned by `sqrt(9)`? 3.0

✔ Correct, the sqrt function returns the square root of its argument.

What are some math functions you could see yourself using or would like to use in a program?

Some functions I can see myself using or would like to use is the power, square root, cosine, sine, tangent, and round functions

Resubmit

MR distance between two points

Click this link to access a program that reflects the above steps. Run the program to see how it works.

What questions or tips for others do you have regarding the calculation of distance between two points?

What can be done to make the code more efficient and do what I want it to do exactly?

Resubmit

✔ Thanks for sharing.

Page 13 Program specific functions

Hotspot function definition

All C++ programs have one and only one main function. We have been focusing on defining what to do in the `main` function. Here is a simple `main` function.

```
int main() {
  cout << "Hello World!" << endl;
  return 0;
}
```

In the function header, int specifies the return type of the `main` function.

✔ Correct, the return type goes before the function name.

This main function accepts 0 parameter(s).

✔ Correct, there is nothing inside the pair of parentheses after the function name.

Using the same approach described in the video, we can define the following function to convert miles into kilometers.

```
double mile_to_km(double miles) {
  const double KM_PER_MILE = 1.60934;
  double km = miles * KM_PER_MILE;
  return km;
}
```

What is the return type of the `mile_to_km` function? double

✔ Correct, the return type of a function is listed before the function name.

How many parameter(s) does the `mile_to_km` function take? 1

✔ Correct! The function expects one parameter.

In your own word, what is the purpose of setting up the miles parameter in the `mile_to_km` function?

In order to have a set variable to use in order to use to calculate your goal of miles to kilometer.

Resubmit

✔ Thanks for sharing.

LBD function calls

```
square(2 * 5)
```

Suppose we make the above function call.

What value would the parameter `number` receive? 10

What value would be stored in the local variable `squared` before the return statement? 100

✔ Correct, number will receive the result of the multiplication as its value.

✔ Correct, squared would take the result of number * number.

```
cout << square(2 * 5);
```

What would the above statement display? 100

✔ Correct! The return statement in the `square` function sends the value of `squared` back as the result of the function call.

Now let's consider the mile_to_km function.

```
double mile_to_km(double miles) {
  const double KM_PER_MILE = 1.60934;
  double km = miles * KM_PER_MILE;
  return km;
}
```

```
mile_to_km(100)
```

Suppose we make the above function call.

What value would the parameter `miles` receive? 100

What value would be stored in the local variable `km` before the return statement? 160.934

✔ Correct, the function call assigns 100 as the value for miles.

✔ Correct, km receives the result of miles * KM_PER_MILE.

```
cout << mile_to_km(100);
```

What would the above statement display? 160.934

✔ Correct! The return statement in the `mile_to_km` function sends the value of `km` back as the result of the function call.

LBD scopes

Mark all of the following that are local to the `mile_to_km` function.

- [ ] miles_per_hour
- [x] km
- [x] miles

Check My Answer

✔ Correct! Only these two variable are declared in this function.

Mark all of the following that are local to the `main` function.

- [x] miles_per_hour
- [ ] miles
- [ ] km

Check My Answer

✔ Correct, only miles_per_hour is declared in main.

Add the following statement in the `mile_to_km` function to see the error message generated by the compiler.

```
cout << miles_per_hour;
```

Remove the above statement but add the following statement in the `main` function to see the error message generated by the compiler.

```
cout << miles;
```

In your own words, explain why the compiler is not happy with the above experiment.

It's unhappy with the first one because miles_per_hour has not yet been defined and it's unhappy with the second one because of

Resubmit

✔ Thanks for sharing.

MR debugging strategies

As mentioned in the video, "everyone gets bugs". What debugging strategies have your found helpful?

What I found helpful were the 4 steps of : Describing your problem, hunting for bugs, trying out solutions and then documenting each bug.

Resubmit

✔ Thanks for sharing.

Page 15 The char data type

LBD ASCII binary for upper and lower case letters

The letter 'T' has `1010100` as its binary ASCII value. Which of the following is the binary ASCII value for the letter 't'?

- 1010100
- 1110100
- 1010101

Correct; the difference between the ASCII values of 'T' and 't' is at their 32s bit of the binary code.

The letter 'q' has `1110001` as its binary ASCII value. Which of the following is the binary ASCII value for the letter 'Q'?

- 1110000
- 1110001
- 1010001

Correct; the difference between the ASCII values of 'Q' and 'q' is at their 32s bit of the binary code.

Hotspot declare char variable

Describe a scenario when it would be beneficial to collect a single character from users.

When creating some sort of test, survey, or anything that has multiple options as a response.

Resubmit

Thanks for sharing.

LBD type casting char and int

As a reminder, the ASCII value for uppercase letters are ranged **consecutively**. That is, the ASCII value is 65 for 'A', 66 for 'B', ..., 90 for 'Y', and 91 for 'Z'.

```
int diff = 'Z' - 'X';
```

What would be the value stored in `diff` after the above statement? 2

✔ Correct! The letter 'X' is two letters before 'Z'

```
char letter = 66;
cout << letter;
```

What would be displayed be the above output statement? B

✔ Correct! The ASCII value of 66 is for the letter B.

```
char letter = 'Z' - 2;
cout << letter;
```

What would be displayed be the above output statement? X

✔ Correct! The ASCII value of 66 is for the letter B.

As a reminder, the ASCII value for a lowercase letter is 32 more than its corresponding uppercase letter.

```
char letter = 'D' + 32;
cout << letter;
```

What would be displayed be the above output statement? d

✔ Correct! The ASCII value of the lowercase letter d is 32 more than ASCII value of the uppercase letter D.

```
char letter = 'm' - 32;
cout << letter;
```

What would be displayed be the above output statement? M

✔ Correct! The ASCII value of the uppercase letter M is 32 less than ASCII value of the lower letter m.

Page 16 The string data type

LBD name concatenate

```
string honorific = "Dr.";
string name = "Seuss";
```

Given the above initializations:

What would be the result of `honorific + name`?

- "SeussDr."
- "Dr.Seuss"

✔ Correct!

```
string first_name, last_name, full_name;
cout << "What is your first name?t";
cin >> first_name;
cout << "What is your last name?t";
cin >> last_name;
```

Given the above code segment that declares, prompts for, and collects the first and last name of a user.

In the following space, write an ***assignment statement*** that updates the `full_name` variable based on the data collected in `first_name` and `last_name`. For example, if a user enters "Cesar" as the first name and "Chavez" as the last name, the assignment statement shall store `"Cesar Chavez"` into `full_name`.

```
full_name = first_name + ' '+ last_name
cout << "Your full name is " << full_name << "." <<endl;
```

Resubmit

✔ full_name = first_name + ' ' + last_name;

LBD name expansion

What would be stored in name after the above statements?

- (●) "Dr. Seuss"
- ( ) "Seuss Dr."
- ( ) "Seuss"

✔ Correct! The operation "Dr. " + name generates "Dr. Seuss" before the = operator stores the result into name.

Given the following code segment

```
string str1 = "King";
string str2 = "Arthur";
str1 += str2;
```

What would be the value in str1? "KingArthur"

✔ Correct.

What would be the value in `str2`? "Arthur"

✔ Correct. `str1 += str2` would not change `str2`.

Which of the following is equivalent to str1 += str2;

- (●) str1 = str1 + str2;
- ( ) str2 = str2 + str1;
- ( ) str1 = str2 + str1;

✔ Correct. This will update str1 using the result of str1 + str2.

LBD subscriptor operator

Enter the index so the following would return `','`

number[2]

✔ Correct! Great job start counting from 0.

What letter is noun[4]? ' i '

✔ Correct! Good job starting the index at 0.

What letter is `predicate[2]` ? ' e '

✔ Correct!

Enter the index so the following would return `'a'`

verb[1]

✔ Correct!

Enter the index so the following would return `'d'`

adv[5]

✔ Correct!