Lab Exercise 2 – C++ Build using Bazel

In this lab exercise, you'll create a simple C++ project and build it using Bazel. We'll create a C++ program that calculates the factorial of a number.

Objective: Create a C++ program to calculate the factorial of a number and build it using Bazel.

Steps:

1. Project Setup:

Create a new directory for your project and navigate to it in your terminal:

```
mkdir bazel_cpp_factorial

cd bazel_cpp_factorial
```

2. Source Code:

Inside your project directory, create a C++ source code file named factorial.cpp:

```
// factorial.cpp

#include <iostream>

int factorial(int n) {

    if (n == 0 || n == 1) {

        return 1;

    } else {

        return n * factorial(n - 1);

    }

}

int main() {

    int num = 5;

    std::cout << "Factorial of " << num << " is: " << factorial(num) << std::endl;

    return 0;
}
```

3. Bazel BUILD File:

Create a BUILD file in the same directory as your source code:

```
cc_binary(
  name = "factorial",
  srcs = ["factorial.cpp"],
)
```

This BUILD file defines a C++ binary target named "factorial" that depends on the "factorial.cpp" source file.

4. Create WORKSPACE file:

Create a WORKSPACE file in root directory.

5. Build with Bazel:

- Open your terminal and navigate to your project directory.
- Build the C++ program using Bazel:

```
bazel build :factorial
```

5. Run the Program:

Once the build is complete, you can run the program:

```
bazel-bin/factorial
```

You should see the output, which will be the factorial of the number specified in the factorial.cpp source code.

6. Experiment:

Feel free to modify the factorial.cpp source code or the BUILD file to add more features or create additional C++ targets. Bazel makes it easy to manage and build C++ projects of various sizes and complexities.

This lab exercise demonstrates how to create a simple C++ project and build it using Bazel. You can extend this example by adding more source files, tests, or additional C++ targets to explore Bazel's capabilities further.