Lab Exercise 6.1 – C++ Build using Bazel

(Sharing Variables using .bzl file)

BUILD files are intended to be simple and declarative. They will typically consist of a series of a target declarations. As your code base and your BUILD files get larger, you will probably notice some duplication, such as:

```
cc_library(
  name = "foo",
  copts = ["-DVERSION=5"],
  srcs = ["foo.cc"],
)

cc_library(
  name = "bar",
  copts = ["-DVERSION=5"],
  srcs = ["bar.cc"],
  deps = [":foo"],
)
```

Code duplication in BUILD files is usually fine. This can make the file more readable: each declaration can be read and understood without any context. This is important, not only for humans, but also for external tools. For example, a tool might be able to read and update BUILD files to add missing dependencies. Code refactoring and code reuse might prevent this kind of automated modification.

If it is useful to share values (for example, if values must be kept in sync), you can introduce a variable:

```
COPTS = ["-DVERSION=5"]

cc_library(
```

```
  name = "foo",
  copts = COPTS,
  srcs = ["foo.cc"],
)

cc_library(
  name = "bar",
  copts = COPTS,
  srcs = ["bar.cc"],
  deps = [":foo"],
)
```

Multiple declarations now use the value COPTS. By convention, use uppercase letters to name global constants.

Sharing variables across multiple BUILD files

If you need to share a value across multiple BUILD files, you have to put it in a .bzl file. .bzl files contain definitions (variables and functions) that can be used in BUILD files.

In path/to/variables.bzl, write:

```
COPTS = ["-DVERSION=5"]
```

Then, you can update your BUILD files to access the variable:

```
load("//path/to:variables.bzl", "COPTS")

cc_library(
  name = "foo",
  copts = COPTS,
  srcs = ["foo.cc"],
)
```

```
cc_library(
  name = "bar",
  copts = COPTS,
  srcs = ["bar.cc"],
  deps = [":foo"],
)
```