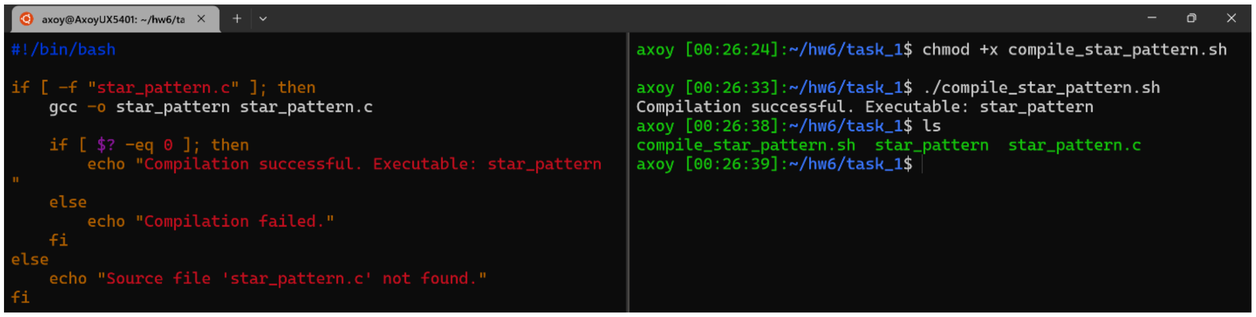
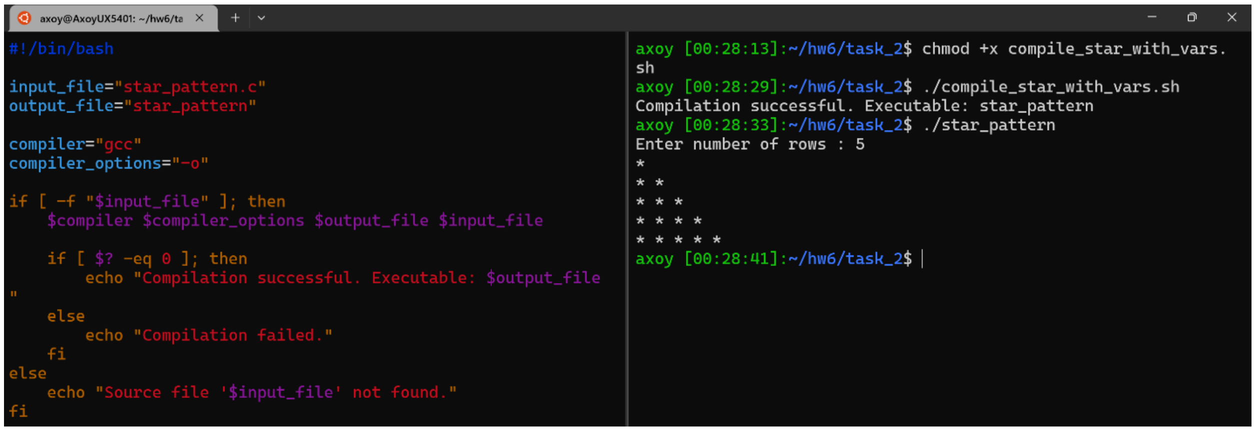
1) Write a compilation shell script to compile “star_pattern.c” using GCC.



```
#!/bin/bash

if [ -f "star_pattern.c" ]; then
    gcc -o star_pattern star_pattern.c

    if [ $? -eq 0 ]; then
        echo "Compilation successful. Executable: star_pattern
"
    else
        echo "Compilation failed."
    fi
else
    echo "Source file 'star_pattern.c' not found."
fi
```

```
axoy [00:26:24]:~/hw6/task_1$ chmod +x compile_star_pattern.sh

axoy [00:26:33]:~/hw6/task_1$ ./compile_star_pattern.sh
Compilation successful. Executable: star_pattern
axoy [00:26:38]:~/hw6/task_1$ ls
compile_star_pattern.sh  star_pattern  star_pattern.c
axoy [00:26:39]:~/hw6/task_1$
```

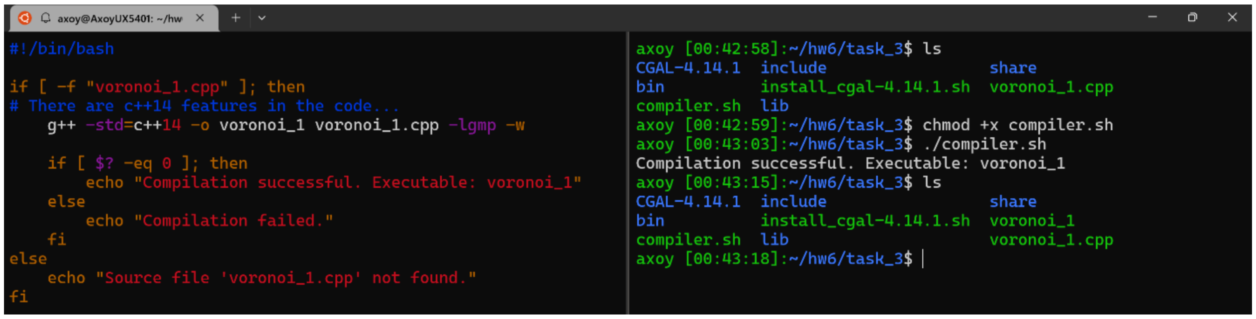
2) For the previous example, use shell variables to represent input filename, output filename, compiler name, and compilation options, and write a more advanced compilation shell script using these variables.



```
#!/bin/bash

input_file="star_pattern.c"
output_file="star_pattern"

compiler="gcc"
compiler_options="-o"

if [ -f "$input_file" ]; then
    $compiler $compiler_options $output_file $input_file

    if [ $? -eq 0 ]; then
        echo "Compilation successful. Executable: $output_file
"
    else
        echo "Compilation failed."
    fi
else
    echo "Source file '$input_file' not found."
fi
```

```
axoy [00:28:13]:~/hw6/task_2$ chmod +x compile_star_with_vars.
sh
axoy [00:28:29]:~/hw6/task_2$ ./compile_star_with_vars.sh
Compilation successful. Executable: star_pattern
axoy [00:28:33]:~/hw6/task_2$ ./star_pattern
Enter number of rows : 5
*
* *
* * *
* * * *
* * * * *
axoy [00:28:41]:~/hw6/task_2$
```

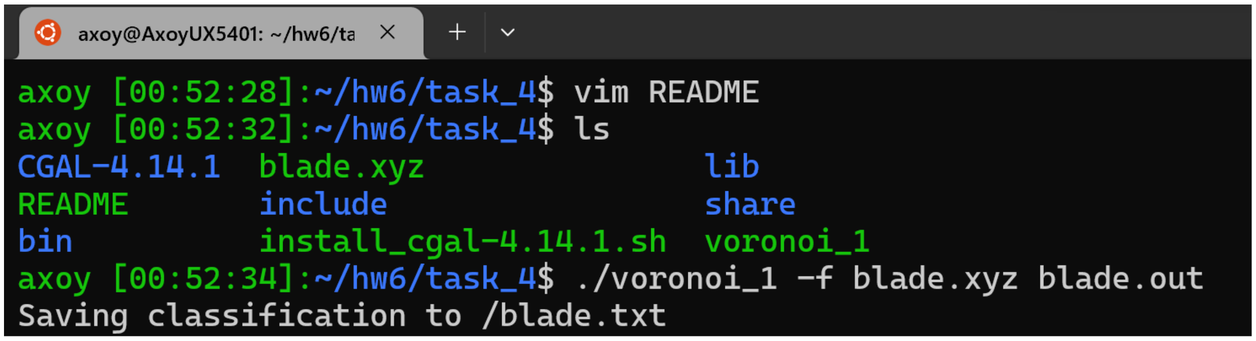
3) Write a shell script to compile “voronoi_1.cpp” into a binary “voronoi_1” using G++ and dynamic linking. This program uses external dependencies which must be installed first (libcgal, libgmp). Use the provided script install_cgal-4.14.1.sh to install CGAL. Use the -std=c++11 option.



```
#!/bin/bash

if [ -f "voronoi_1.cpp" ]; then
# There are c++14 features in the code...
    g++ -std=c++14 -o voronoi_1 voronoi_1.cpp -lgmp -w

    if [ $? -eq 0 ]; then
        echo "Compilation successful. Executable: voronoi_1"
    else
        echo "Compilation failed."
    fi
else
    echo "Source file 'voronoi_1.cpp' not found."
fi
```

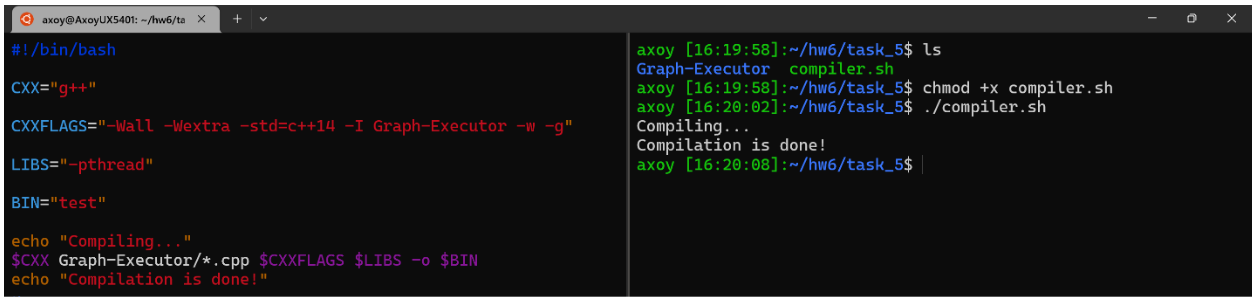
```
axoy [00:42:58]:~/hw6/task_3$ ls
CGAL-4.14.1  include                 share
bin          install_cgal-4.14.1.sh  voronoi_1.cpp
compiler.sh  lib
axoy [00:42:59]:~/hw6/task_3$ chmod +x compiler.sh
axoy [00:43:03]:~/hw6/task_3$ ./compiler.sh
Compilation successful. Executable: voronoi_1
axoy [00:43:15]:~/hw6/task_3$ ls
CGAL-4.14.1  include                 share
bin          install_cgal-4.14.1.sh  voronoi_1
compiler.sh  lib                     voronoi_1.cpp
axoy [00:43:18]:~/hw6/task_3$
```

4) Provide a one-liner command to run the compiled binary executable "voronoi_1" (mind the need to export the directory with CGAL shared library



```
axoy [00:52:28]:~/hw6/task_4$ vim README
axoy [00:52:32]:~/hw6/task_4$ ls
CGAL-4.14.1  blade.xyz              lib
README       include                share
bin          install_cgal-4.14.1.sh  voronoi_1
axoy [00:52:34]:~/hw6/task_4$ ./voronoi_1 -f blade.xyz blade.out
Saving classification to /blade.txt
```

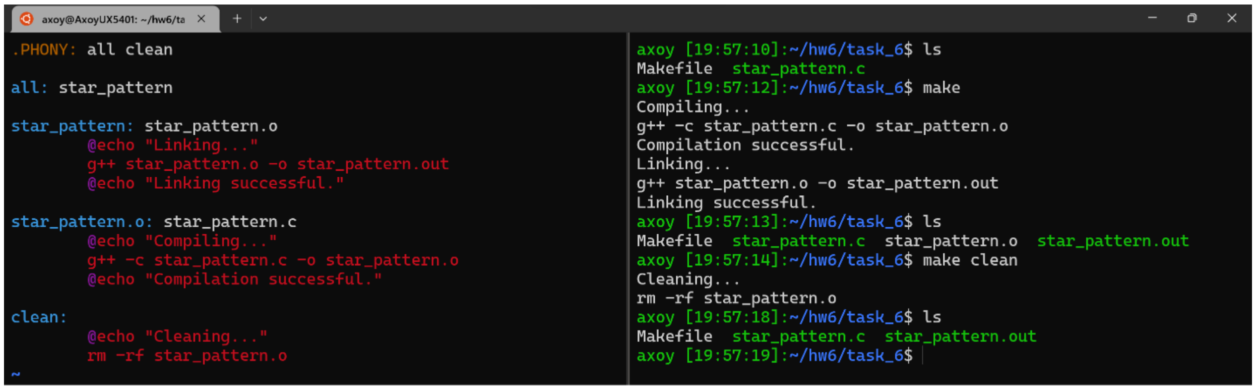
5) Write a compilation script (with shell variables) to compile the project "Graph-Executor" using G++. The target executable is test, all other files compile to non-executable objects. Use these options (search the GCC docs to find the correct flag): a. Enable all warnings about questionable constructions b. Enable extra warning flags c. Compile for c++14 language standard d. Generate debug information e. When linking test, additionally use -pthread



```
#!/bin/bash

CXX="g++"

CXXFLAGS="-Wall -Wextra -std=c++14 -I Graph-Executor -w -g"

LIBS="-pthread"

BIN="test"

echo "Compiling..."
$CXX Graph-Executor/*.cpp $CXXFLAGS $LIBS -o $BIN
echo "Compilation is done!"
```

```
axoy [16:19:58]:~/hw6/task_5$ ls
Graph-Executor  compiler.sh
axoy [16:19:58]:~/hw6/task_5$ chmod +x compiler.sh
axoy [16:20:02]:~/hw6/task_5$ ./compiler.sh
Compiling...
Compilation is done!
axoy [16:20:08]:~/hw6/task_5$
```

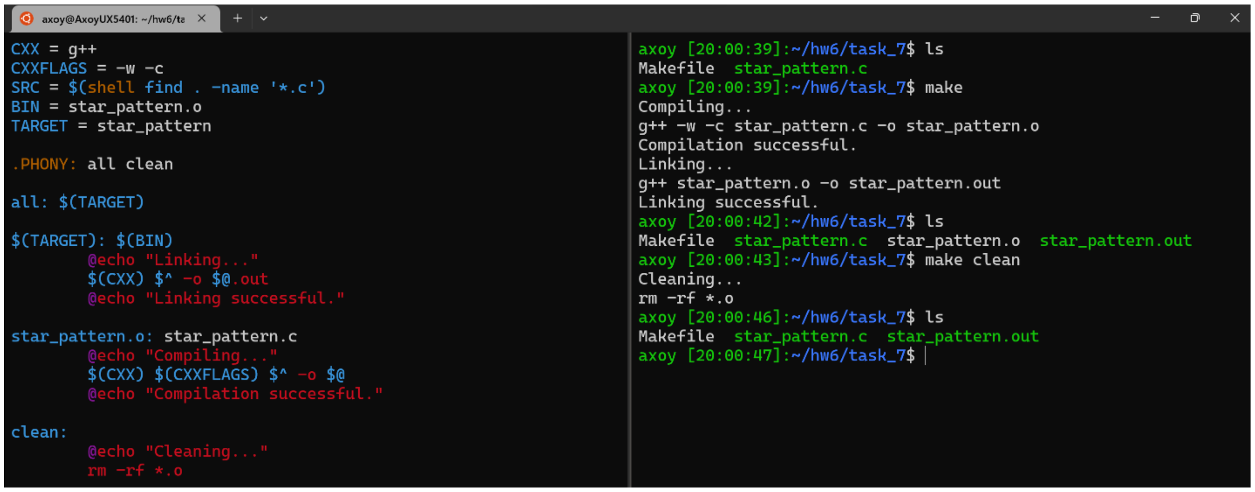
6) Write a Makefile to compile "star_pattern.c" using GCC. Provide an "all" target to build the program an a "clean" target to remove the built binary.



```
.PHONY: all clean

all: star_pattern

star_pattern: star_pattern.o
        @echo "Linking..."
        g++ star_pattern.o -o star_pattern.out
        @echo "Linking successful."

star_pattern.o: star_pattern.c
        @echo "Compiling..."
        g++ -c star_pattern.c -o star_pattern.o
        @echo "Compilation successful."

clean:
        @echo "Cleaning..."
        rm -rf star_pattern.o
~
```

```
axoy [19:57:10]:~/hw6/task_6$ ls
Makefile  star_pattern.c
axoy [19:57:12]:~/hw6/task_6$ make
Compiling...
g++ -c star_pattern.c -o star_pattern.o
Compilation successful.
Linking...
g++ star_pattern.o -o star_pattern.out
Linking successful.
axoy [19:57:13]:~/hw6/task_6$ ls
Makefile  star_pattern.c  star_pattern.o  star_pattern.out
axoy [19:57:14]:~/hw6/task_6$ make clean
Cleaning...
rm -rf star_pattern.o
axoy [19:57:18]:~/hw6/task_6$ ls
Makefile  star_pattern.c  star_pattern.out
axoy [19:57:19]:~/hw6/task_6$
```

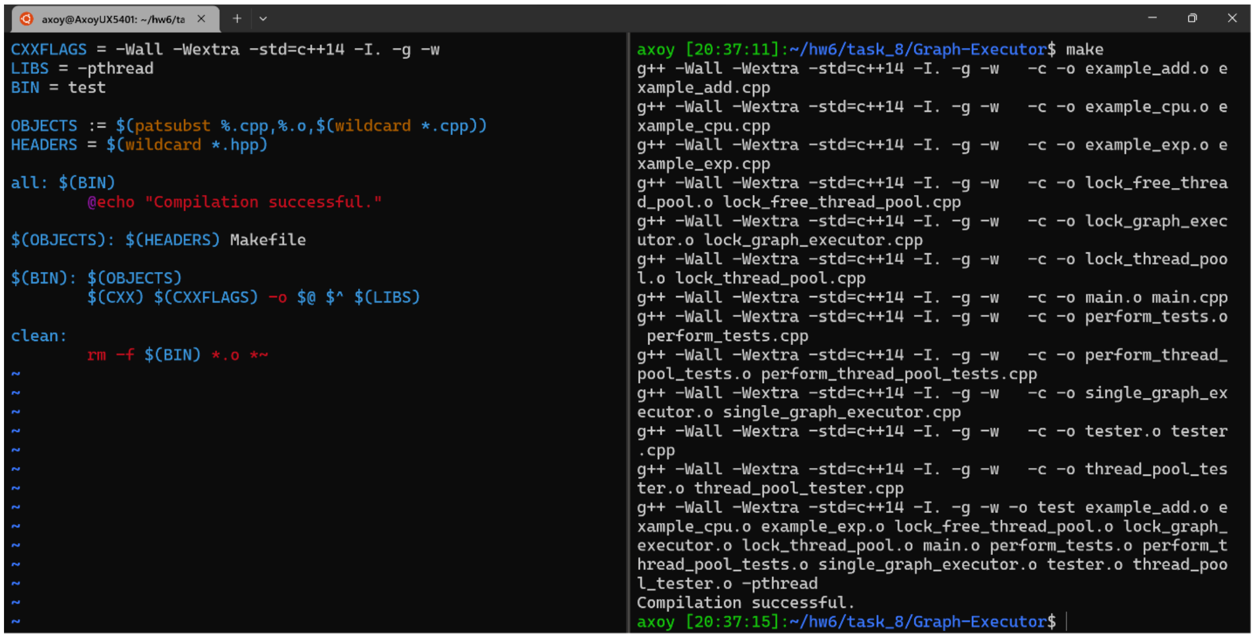
7) For the previous example, write a Makefile with variables to represent compiler name and compilation flags and wildcards to represent input/output filenames.



```
CXX = g++
CXXFLAGS = -w -c
SRC = $(shell find . -name '*.c')
BIN = star_pattern.o
TARGET = star_pattern

.PHONY: all clean

all: $(TARGET)

$(TARGET): $(BIN)
        @echo "Linking..."
        $(CXX) $^ -o $@.out
        @echo "Linking successful."

star_pattern.o: star_pattern.c
        @echo "Compiling..."
        $(CXX) $(CXXFLAGS) $^ -o $@
        @echo "Compilation successful."

clean:
        @echo "Cleaning..."
        rm -rf *.o
```

```
axoy [20:00:39]:~/hw6/task_7$ ls
Makefile  star_pattern.c
axoy [20:00:39]:~/hw6/task_7$ make
Compiling...
g++ -w -c star_pattern.c -o star_pattern.o
Compilation successful.
Linking...
g++ star_pattern.o -o star_pattern.out
Linking successful.
axoy [20:00:42]:~/hw6/task_7$ ls
Makefile  star_pattern.c  star_pattern.o  star_pattern.out
axoy [20:00:43]:~/hw6/task_7$ make clean
Cleaning...
rm -rf *.o
axoy [20:00:46]:~/hw6/task_7$ ls
Makefile  star_pattern.c  star_pattern.out
axoy [20:00:47]:~/hw6/task_7$
```

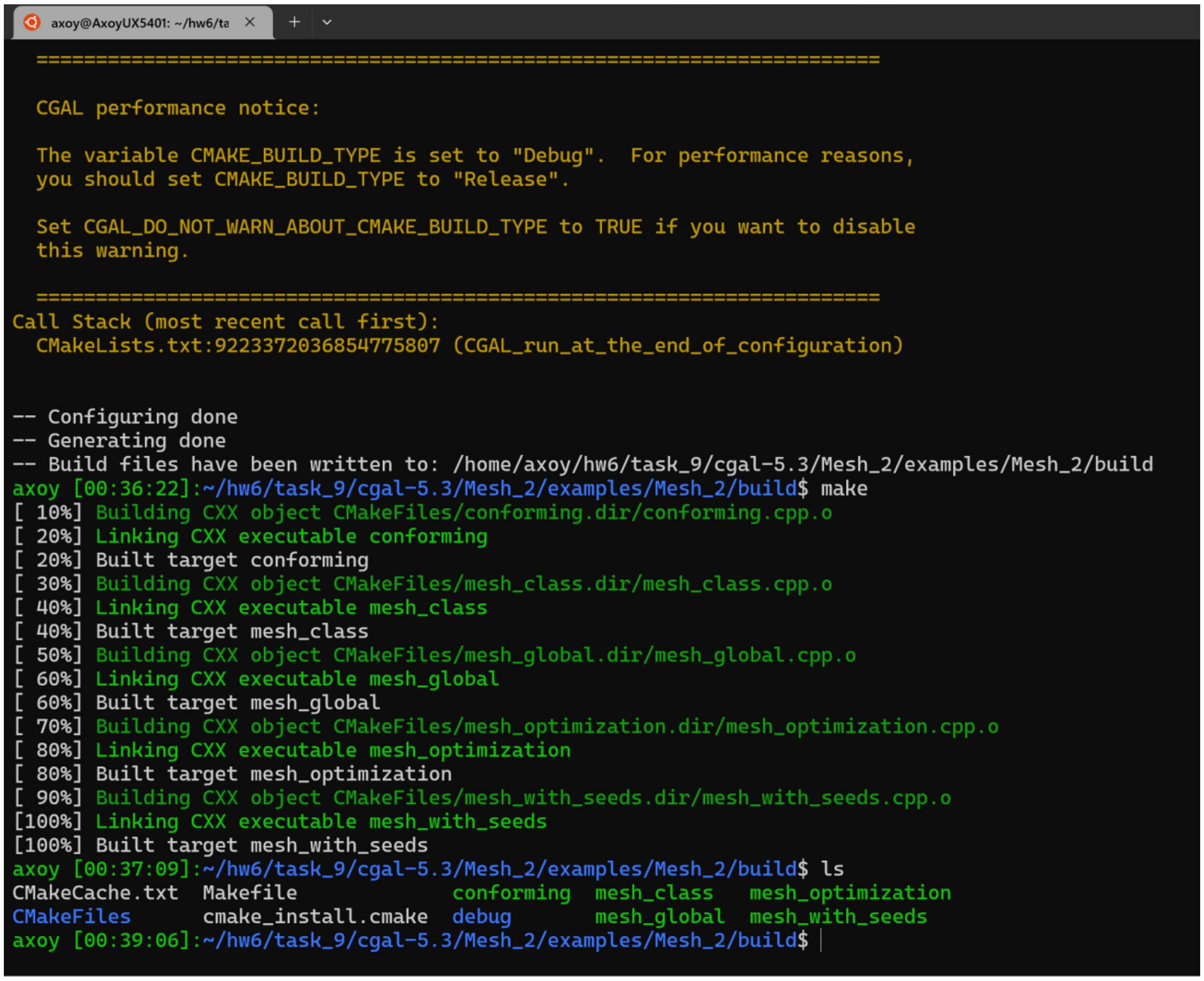
8) Write a Makefile (with shell variables, wildcards,) to compile the project "Graph-Executor" using G++. The target executable is test, all other files compile to non-executable objects. Use the same compilation options as previously.



```
CXXFLAGS = -Wall -Wextra -std=c++14 -I. -g -w
LIBS = -pthread
BIN = test

OBJECTS := $(patsubst %.cpp,%.o,$(wildcard *.cpp))
HEADERS = $(wildcard *.hpp)

all: $(BIN)
        @echo "Compilation successful."

$(OBJECTS): $(HEADERS) Makefile

$(BIN): $(OBJECTS)
        $(CXX) $(CXXFLAGS) -o $@ $^ $(LIBS)

clean:
        rm -f $(BIN) *.o *~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

```
axoy [20:37:11]:~/hw6/task_8/Graph-Executor$ make
g++ -Wall -Wextra -std=c++14 -I. -g -w   -c -o example_add.o e
xample_add.cpp
g++ -Wall -Wextra -std=c++14 -I. -g -w   -c -o example_cpu.o e
xample_cpu.cpp
g++ -Wall -Wextra -std=c++14 -I. -g -w   -c -o example_exp.o e
xample_exp.cpp
g++ -Wall -Wextra -std=c++14 -I. -g -w   -c -o lock_free_threa
d_pool.o lock_free_thread_pool.cpp
g++ -Wall -Wextra -std=c++14 -I. -g -w   -c -o lock_graph_exec
utor.o lock_graph_executor.cpp
g++ -Wall -Wextra -std=c++14 -I. -g -w   -c -o lock_thread_poo
l.o lock_thread_pool.cpp
g++ -Wall -Wextra -std=c++14 -I. -g -w   -c -o main.o main.cpp
g++ -Wall -Wextra -std=c++14 -I. -g -w   -c -o perform_tests.o
 perform_tests.cpp
g++ -Wall -Wextra -std=c++14 -I. -g -w   -c -o perform_thread_
pool_tests.o perform_thread_pool_tests.cpp
g++ -Wall -Wextra -std=c++14 -I. -g -w   -c -o single_graph_ex
ecutor.o single_graph_executor.cpp
g++ -Wall -Wextra -std=c++14 -I. -g -w   -c -o tester.o tester
.cpp
g++ -Wall -Wextra -std=c++14 -I. -g -w   -c -o thread_pool_tes
ter.o thread_pool_tester.cpp
g++ -Wall -Wextra -std=c++14 -I. -g -w -o test example_add.o e
xample_cpu.o example_exp.o lock_free_thread_pool.o lock_graph_
executor.o lock_thread_pool.o main.o perform_tests.o perform_t
hread_pool_tests.o single_graph_executor.o tester.o thread_poo
l_tester.o -pthread
Compilation successful.
axoy [20:37:15]:~/hw6/task_8/Graph-Executor$
```

9) Use CMake to build the Mesh_2 example in the CGAL library: a. Download the latest CGAL from https://github.com/CGAL/cgal/archive/refs/tags/v5.3.tar.gz using wget b. Extract the archive contents (hint: use tar -xzvf) c. Follow the instructions in /INSTALL.md to build the code in /Mesh_2/examples/Mesh_2 directory using CMake and make



```
  ======================================================================

  CGAL performance notice:

  The variable CMAKE_BUILD_TYPE is set to "Debug".  For performance reasons,
  you should set CMAKE_BUILD_TYPE to "Release".

  Set CGAL_DO_NOT_WARN_ABOUT_CMAKE_BUILD_TYPE to TRUE if you want to disable
  this warning.

  ======================================================================
Call Stack (most recent call first):
  CMakeLists.txt:9223372036854775807 (CGAL_run_at_the_end_of_configuration)


-- Configuring done
-- Generating done
-- Build files have been written to: /home/axoy/hw6/task_9/cgal-5.3/Mesh_2/examples/Mesh_2/build
axoy [00:36:22]:~/hw6/task_9/cgal-5.3/Mesh_2/examples/Mesh_2/build$ make
[ 10%] Building CXX object CMakeFiles/conforming.dir/conforming.cpp.o
[ 20%] Linking CXX executable conforming
[ 20%] Built target conforming
[ 30%] Building CXX object CMakeFiles/mesh_class.dir/mesh_class.cpp.o
[ 40%] Linking CXX executable mesh_class
[ 40%] Built target mesh_class
[ 50%] Building CXX object CMakeFiles/mesh_global.dir/mesh_global.cpp.o
[ 60%] Linking CXX executable mesh_global
[ 60%] Built target mesh_global
[ 70%] Building CXX object CMakeFiles/mesh_optimization.dir/mesh_optimization.cpp.o
[ 80%] Linking CXX executable mesh_optimization
[ 80%] Built target mesh_optimization
[ 90%] Building CXX object CMakeFiles/mesh_with_seeds.dir/mesh_with_seeds.cpp.o
[100%] Linking CXX executable mesh_with_seeds
[100%] Built target mesh_with_seeds
axoy [00:37:09]:~/hw6/task_9/cgal-5.3/Mesh_2/examples/Mesh_2/build$ ls
CMakeCache.txt  Makefile            conforming  mesh_class   mesh_optimization
CMakeFiles      cmake_install.cmake debug       mesh_global  mesh_with_seeds
axoy [00:39:06]:~/hw6/task_9/cgal-5.3/Mesh_2/examples/Mesh_2/build$
```