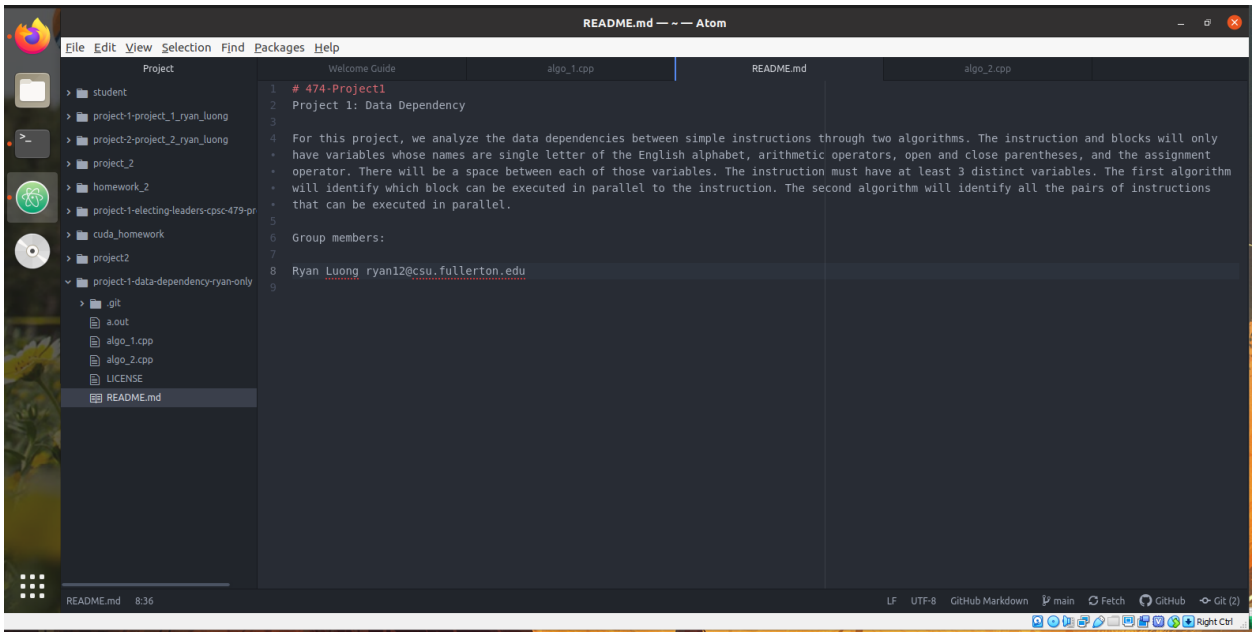
Project 1 : Data Dependency

Group Members:

Ryan Luong ryan12@csu.fullerton.edu



Summary:

For this project, I used C++ to analyze the data dependencies between simple instructions through two algorithms. The instruction and blocks will only have variables whose names are single letters of the English alphabet, arithmetic operators, open and close parentheses, and the assignment operator. There will be a space between each of those variables. The instruction must have at least three distinct variables. The first algorithm will identify which block can be executed parallel to the instruction. The second algorithm will identify all the pairs of instructions that can be executed in parallel.

Algorithm #1 Pseudocode:

```
string instruction
display "Input your instruction: " and get the instruction input from the
user {
    vector<string> instructionList
    split instruction based on spaces and push it to instructionList
    }
for (each element in instruction) {
    if the element size is greater than or equal to 0
```

```
        Terminate the program
        }
vector<char> instructionVariable
for (each element in instructionList) {
    char letter = element
    if the letter is part of the English Alphabet
        push it to instructionVariable
        }
int numOfBlocks = 0
map<char, vector<string>> blocks;
display "Enter the # of blocks: " and get the input from the user
for i in range(0, numOfBlocks.size()) {
    string block
    vector<string> blockList
    display "Block {i}: " and get the block input from the user
    split block based on spaces and push it to blockList
    insert the first element of blocklist as the key and blockList as the
value to blocks
    }
for (key in blocks) {
    for (value in key) {
        if the value is greater than or equal to 2 {
            Terminate the program
        }
        }
        }
map<char, set<char>> inputWithOutput
set<char> instructionVariableOutput
for i in range(i, instructionVariable.size()) {
    Insert instructionVariable[i] to instructionVariableOutput
    }
Insert the first element of instructionVariable as the key and
instructionVariableOutput as the value to inputWithOutput
for (key in blocks) {
    char input = key
    set<char> output
    for (value in key) {
        if the value is part of the English Alphabet
            insert it to output
            }
```

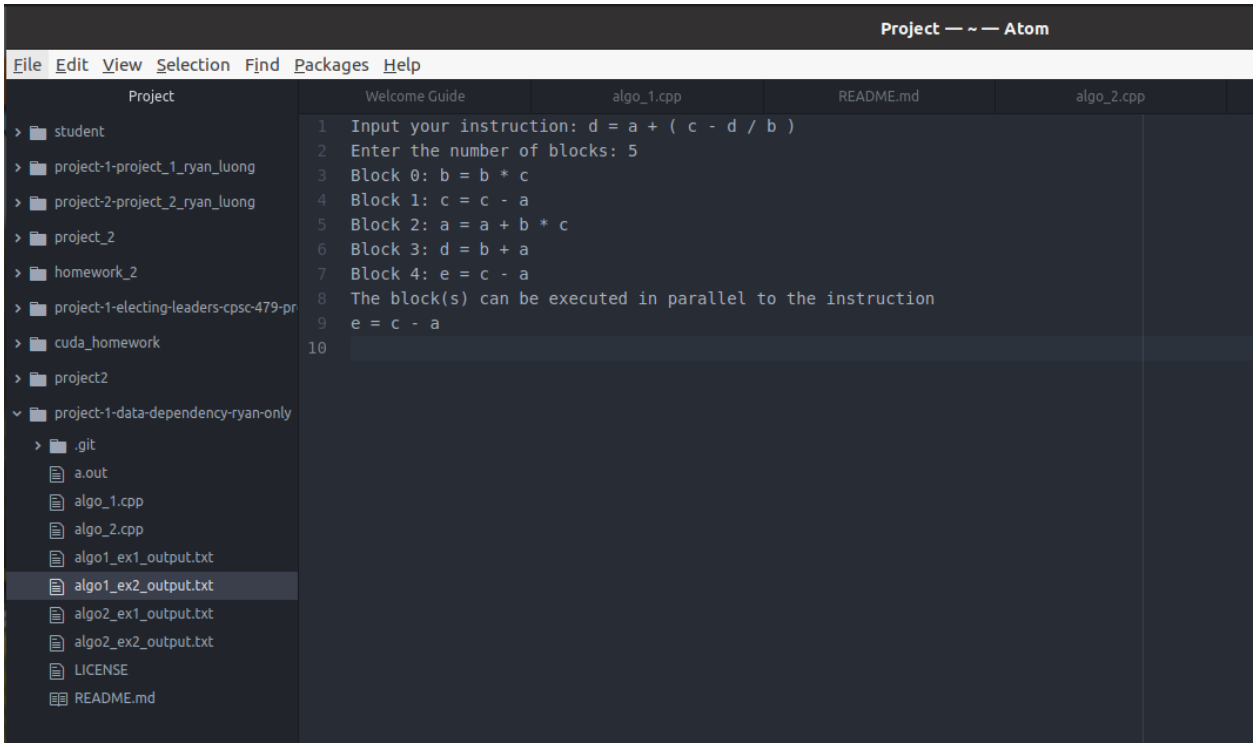
```
    insert input as the key and output as the value to inputWithOutput
    }
vector<char> allInputs
insert the first element of instructionVariable to allInputs
for (key in blocks) {
    insert the key to allInputs
    }
double elapsedTime
vector<vector<string>> results
for i in range(1, allInputs.size()) {
    char first = allInputs[0], second = allInputs[i]
    set<char> flowDependency, antiDependency, outputDependency,
              inputL1 = first, outputL1 = inputWithOutput[first], inputL2
= second,
              outputL2 = inputWithOutput[second]
    start the time
    get the intersection of the output of l1 and the input of l2
    get the intersection of the input of l1 and the output of l2
    get the intersection of input of l1 and l2
    end the time
    add it to elapsedTime
    if the number of blocks is 3 and if the elapsed time is over or equal
to 60 minutes
        terminate the program
    if the 3 intersection return empty
        push blocks[second] to results
    }
if result is empty
    display "NONE"
else
    display the output
```

The code was compiled and run using:

g++ algo_1.cpp
./a.out > algo1_ex1_output.txt
./a.out > algo1_ex2_output.txt

N = 5

```
student@tuffix-vm:~/project-1-data-dependency-ryan-only$ ./a.out > algo1_ex2_output.txt
d = a + ( c - d / b )
5
b = b * c
c = c - a
a = a + b * c
d = b + a
e = c - a
```



Explanation:

The first line is asking for the instruction
The second line is asking the # of blocks
The rest is asking for the block inputs

Algorithm #2 Pseudocode:

```
int numOfBlocks = 0
map<char, vector<string>> blocks;
display "Enter the # of blocks: " and get the input from the user
for i in range(0, numOfBlocks) {
    string block
    vector<string> blockList
    display "Block {i}: " and get the block input from the user
```

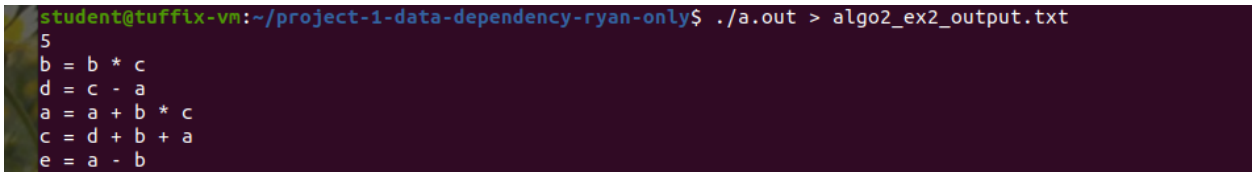
```
    split block based on spaces and push it to blockList
    insert the first element of blocklist as the key and blockList as the
value to blocks
    }
for (key in blocks) {
    for (value in key) {
        if the value is greater than or equal to 2 {
            Terminate the program
        }
        }
        }
map<char, set<char>> inputWithOutput
for (key in blocks) {
    char input = key
    set<char> output
    for (value in key) {
        if the value is part of the English Alphabet
            insert it to output
            }
    insert input as the key and output as the value to inputWithOutput
    }
vector<char> allInputs
for (key in blocks) {
    insert the key to allInputs
    }
double elaspedTime
vector<vector<string>> results
for i in range(1, allInputs.size()) {
    char first = allInputs[0], second = allInputs[i]
    set<char> flowDependency, antiDependency, outputDependency,
              inputL1 = first, outputL1 = inputWithOutput[first], inputL2
= second,
              outputL2 = inputWithOutput[second]
    start the time
    get the intersection of output of l1 and input of input of l2
    get the intersection of input of l1 and output of l2
    get the intersection of input of l1 and l2
    end the time
    add it to elapsedTime
```

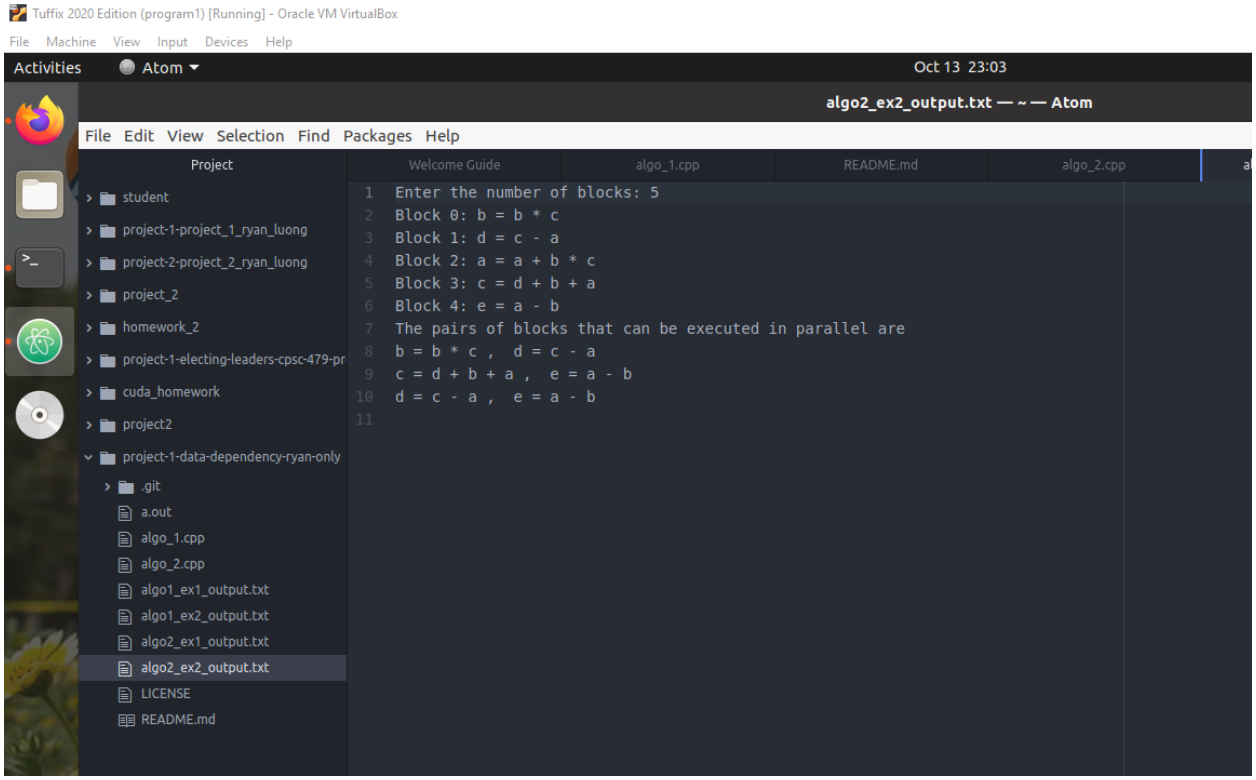
```
    if the number of blocks is 3 and if the elapsed time is over or equal
to 60 minutes
        terminate the program
    if the 3 intersection return empty
        push blocks[second] to results
    }
if result is empty
    display "NONE"
else
    display the output
```

The code was compiled and run using:

g++ algo_2.cpp
./a.out > algo2_ex1_output.txt
./a.out > algo2_ex2_output.txt

N = 5





Explanation:

The first line is asking the # of blocks
The rest is asking for the block inputs