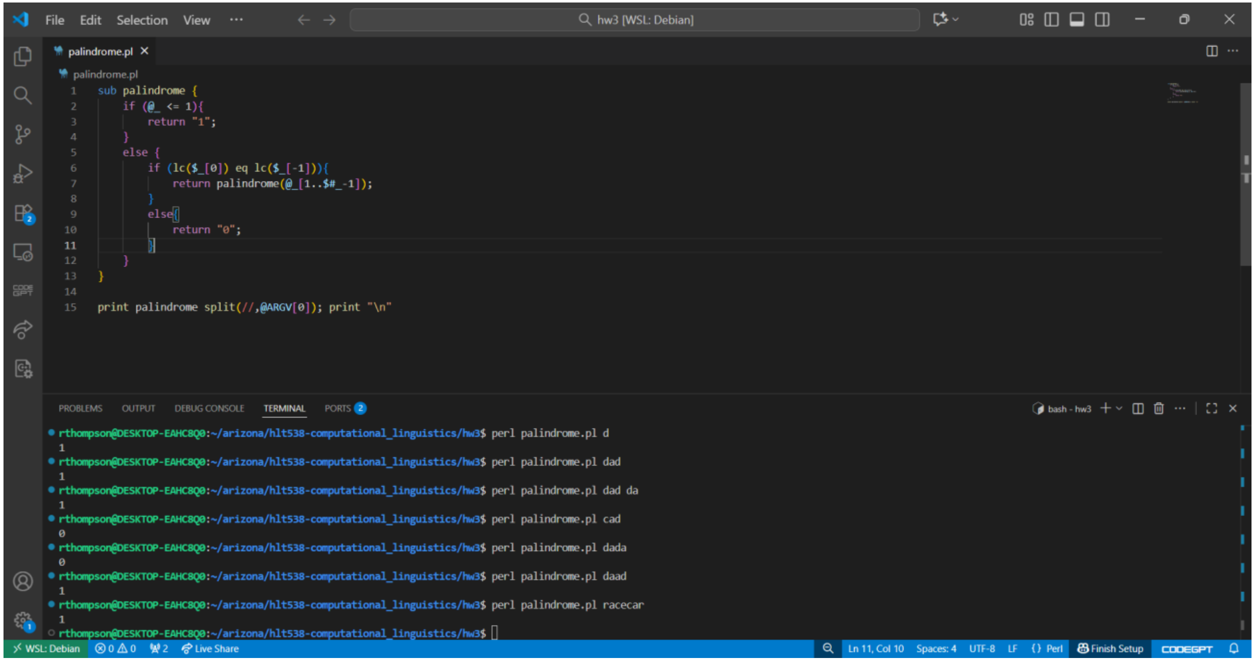
Screenshot for part one. The only meaningful difference between my interpretation and the python script is I elected to convert the string input into an array as I pass it into the palindrome detecting function because I felt like dealing with indexes more than I felt like dealing with the substring function.



File Edit Selection View ···
hw3 [WSL: Debian]
palindrome.pl
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 2
bash - hw3
WSL: Debian ⊗ 0 ⚠ 0 2 Live Share
Ln 11, Col 10 Spaces: 4 UTF-8 LF {} Perl Finish Setup CODEGPT

```
sub palindrome {
    if (@_ <= 1){
        return "1";
    }
    else {
        if (lc($_[0]) eq lc($_[-1])){
            return palindrome(@_[1..$#_-1]);
        }
        else{
            return "0";
        }
    }
}

print palindrome split(//,@ARGV[0]); print "\n"
```

```
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ perl palindrome.pl d
1
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ perl palindrome.pl dad
1
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ perl palindrome.pl dad da
1
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ perl palindrome.pl cad
0
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ perl palindrome.pl dada
0
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ perl palindrome.pl daad
1
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ perl palindrome.pl racecar
1
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ 
```

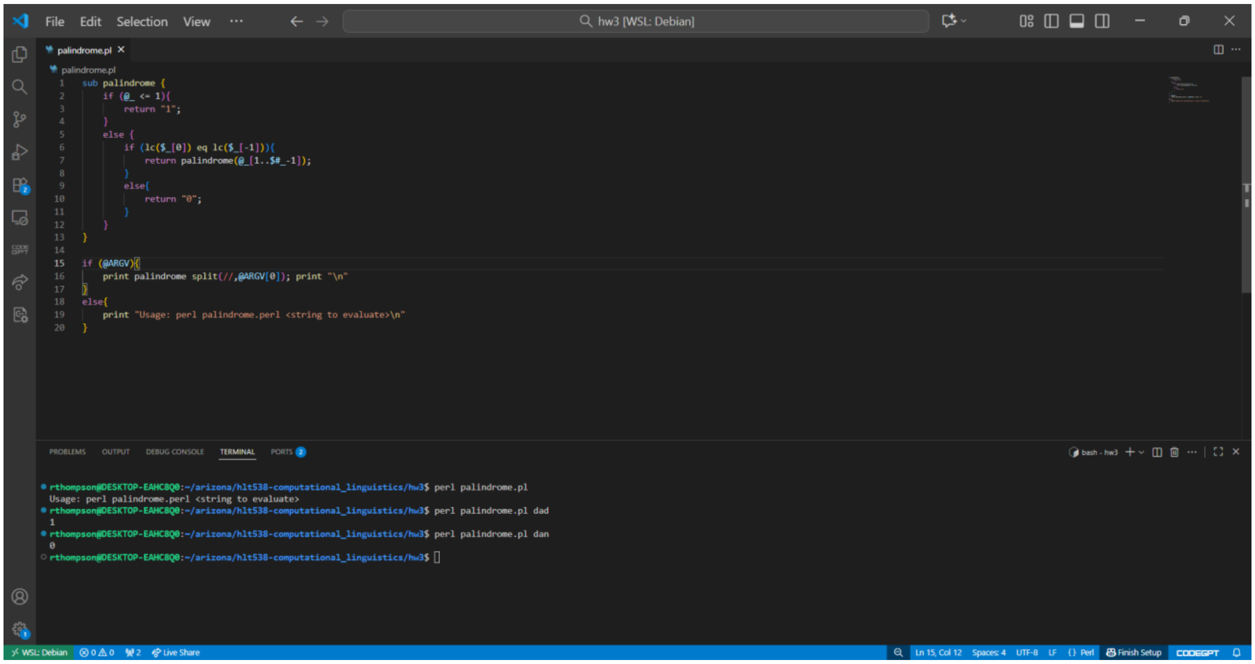
Perl code’’’

```
sub palindrome {
  if (@_ <= 1){
    return "1";
  }
  else {
    if (lc($_[0]) eq lc($_[-1])){
      return palindrome(@_[1..$#_-1]);
    }
    else{
      return "0";
    }
  }
}

print palindrome split(//,@ARGV[0]); print "\n"
```

‘’’

Part 2 screenshot. The only code modified is in the print statements at the bottom. Checked the @ARGV array as a scalar value within an if statement to see if it’s empty or not

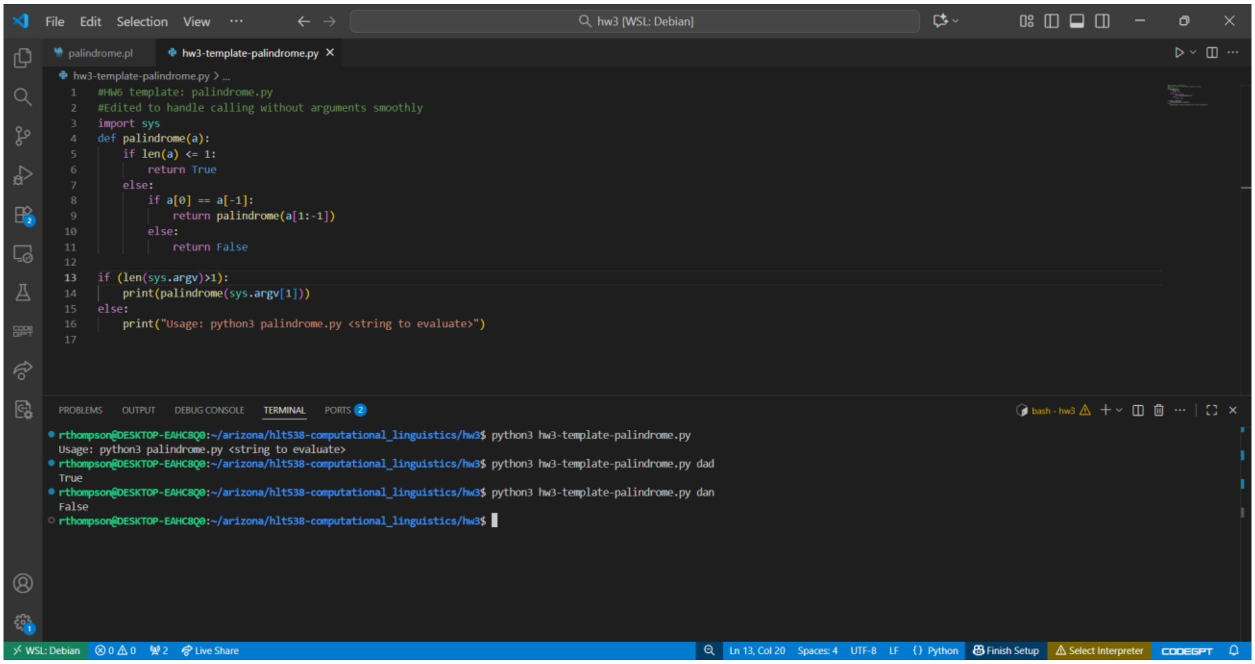


```perl
sub palindrome {
    if (@_ <= 1){
        return "1";
    }
    else {
        if (lc($_[0]) eq lc($_[-1])){
            return palindrome(@_[1..$#_-1]);
        }
        else{
            return "0";
        }
    }
}

if (@ARGV){
    print palindrome split(//,@ARGV[0]); print "\n"
}
else{
    print "Usage: perl palindrome.perl <string to evaluate>\n"
}
```

```
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ perl palindrome.pl
Usage: perl palindrome.perl <string to evaluate>
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ perl palindrome.pl dad
1
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ perl palindrome.pl dan
0
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$
```

Part 3:



```python
#HW6 template: palindrome.py
#Edited to handle calling without arguments smoothly
import sys
def palindrome(a):
    if len(a) <= 1:
        return True
    else:
        if a[0] == a[-1]:
            return palindrome(a[1:-1])
        else:
            return False

if (len(sys.argv)>1):
    print(palindrome(sys.argv[1]))
else:
    print("Usage: python3 palindrome.py <string to evaluate>")
```

```
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ python3 hw3-template-palindrome.py
Usage: python3 palindrome.py <string to evaluate>
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ python3 hw3-template-palindrome.py dad
True
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$ python3 hw3-template-palindrome.py dan
False
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw3$
```