Week 8

Question 1: The Unix nl command prints the lines of a text file with a line number at the start of each line. (It can be useful when printing out programs for dry runs or white-box testing). Write an implementation of this command. It should take the name of the files as a command-line argument.

```
import sys


def nl_command(filename):
    try:
        with open(filename, 'r') as file:
            lines = file.readlines()


        for line_number, line in enumerate(lines, start=1):
            # Print line number and content, right-justified for uniformity
            print(f"{line_number:>6}\t{line.rstrip()}")


    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
    except Exception as e:
        print(f"An error occurred: {e}")


if len(sys.argv) != 2:
    print("Usage: python nl_command.py <filename>")
else:
  nl_command(sys.argv[1])
```

```
C:\Users\karna>python nl_command.py nl.txt
     1  First line
     2  Second line
     3  Third line
```

Question 2: The Unix diff command compares two files and reports the differences, if any.

Write a simple implementation of this that takes two file names as command-line arguments and reports whether or not the two files are the same. (Define "same" as having the same contents.)

```
import sys
import os

def create_file_if_missing(filename, content):
    if not os.path.exists(filename):
        with open(filename, 'w') as file:
            file.write(content)
        print(f"File '{filename}' has been created with default content.")

def diff_command(file1, file2):

    try:
        create_file_if_missing(file1, "Default content for file 1.\n")
        create_file_if_missing(file2, "Default content for file 2.\n")

        with open(file1, 'r') as f1, open(file2, 'r') as f2:
            content1 = f1.readlines()
            content2 = f2.readlines()

        if content1 == content2:
            print("The files are the same.")
        else:
            print("The files are different.")
            max_lines = max(len(content1), len(content2))
            for i in range(max_lines):
                line1 = content1[i].rstrip() if i < len(content1) else "<no line>"
                line2 = content2[i].rstrip() if i < len(content2) else "<no line>"
                if line1 != line2:
```

```
            print(f"Line {i + 1}:\nFile 1: {line1}\nFile 2: {line2}")

    except Exception as e:
        print(f"An error occurred: {e}")

if len(sys.argv) != 3:
    print("Usage: python diff_command.py <file1> <file2>")
else:
    diff_command(sys.argv[1], sys.argv[2])
```

```
C:\Users\karna>python diff_command.py diff1.txt diff2.txt
File 'diff1.txt' has been created with default content.
File 'diff2.txt' has been created with default content.
The files are different.
Line 1:
File 1: Default content for file 1.
File 2: Default content for file 2.
```

```
C:\Users\karna>python diff_command.py diff1.txt diff2.txt
The files are different.
Line 2:
File 1: 1. Easy to learn and use.
File 2: 1. Beginner-friendly syntax.
Line 3:
File 1: 2. Supports multiple programming paradigms.
File 2: 2. Multiparadigm support.
Line 4:
File 1: 3. Extensive standard library.
File 2: 3. Large community and ecosystem.
Line 5:
File 1: 4. Dynamic typing and garbage collection.
File 2: 4. Automatic memory management.
```

```
C:\Users\karna>python diff_command.py diff1.txt diff2.txt
The files are the same.
```

Question 3: The Unix grep command searches a file and outputs the lines in the file that contain a certain pattern. Write an implementation of this. It will take two command-line arguments: the first is the string to look for, and the second is the

file name. The output should be the lines in the file that contain the string.

```
import sys


def grep_command(pattern, filename):
    try:
        with open(filename, 'r') as file:
            lines = file.readlines()


        found = False
        for line_number, line in enumerate(lines, start=1):
            if pattern in line:
                print(f"Line {line_number}: {line.strip()}")
                found = True


        if not found:
            print(f"No lines found containing the pattern: '{pattern}'")


    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")
    except Exception as e:
        print(f"An error occurred: {e}")


if len(sys.argv) != 3:
    print("Usage: python grep_command.py <pattern> <filename>")
else:
    grep_command(sys.argv[1], sys.argv[2])
```

```
C:\Users\karna>python grep_command.py Python grep.txt
Line 2: Python is a versatile programming language.
```

Question 4: The Unix wc command counts the number of lines, words, and characters in a file.

Write an implementation of this that takes a file name as a command-line argument, and then prints the number of lines and characters.

Note: Linux (and Mac) users can use the "wc" command to check the results of their implementation.

```python
import sys

def count_lines_and_characters(filename):
 try:
  with open(filename, 'r', encoding='utf-8') as file:
   lines = file.readlines()
   num_lines = len(lines)
   num_characters = sum(len(line) for line in lines)

   print(f"Lines: {num_lines}")
   print(f"Characters: {num_characters}")
 except FileNotFoundError:
  print(f"Error: The file '{filename}' does not exist.")
 except Exception as e:
  print(f"Error: {e}")

if len(sys.argv)!= 2:
 print("Usage: python wc.py <filename>")
else:
 filename = sys.argv[1]
 count_lines_and_characters(filename)
```

```
This file is created to run the word count command
```

```
nalini@nalini-VirtualBox:~$ echo -e "This file is created to run the word count
command" > wc.txt
nalini@nalini-VirtualBox:~$ nano wc.py
nalini@nalini-VirtualBox:~$ python3 wc.py wc.txt
Lines: 1
Characters: 51
```

Question 5: The Unix spell command is a simple spell-checker. It prints out all the words in a text file that are not found in a dictionary. Write and test an implementation of this, that takes a file name as a command-line argument.

Note: You may want to simplify the program at first by testing with a text file that does not contain any punctuation. A complete version should obviously be able to handle normal files, with punctuation.

Another Note: You will need a list of valid words. Linux users will already have one (probably in /usr/share/dict/words). It is more complicated, as usual, for Windows users. Happily, there are several available on GitHub.

```
import sys
import string

def load_dict(dict_file):
 try:
  with open(dict_file, 'r', encoding='utf-8') as f:
   valid_words = set(f.read().splitlines())
  return valid_words
 except FileNotFoundError:
  print(f"Error: The dictionary file'{dict_file}' was not found."
  sys.exit(1)

def clean_word(word):
 return word.strip(string.punctuation).lower()

def spell_check(spell_cmd, dict_file):
 valid_words = load_dict(dict_file)

 try:
  with open(spell_cmd, 'r', encoding='utf-8') as file:
   words = file.read().split()
   misspelled_words = []

   for word in words:
    cleaned_word = clean_word(word)
    if cleaned_word and cleaned_word not in valid_words:
     misspelled_words.append(word)

   if misspelled_words:
    print("Misspelled words: ")
    for word in misspelled_words:
     print(word)
   else:
    print("No misspelled words found.")
 except FileNotFoundError:
  print(f"Error: The file '{spell_cmd}' was not found")
  sys.exit(1)

if len(sys.argv)!=3:
 print("Usage: python spell_cmd.py <spell_cmd> <dict_file>")

else:
 spell_cmd= sys.argv[1]
 dict_file = sys.argv[2]
 spell_check(spell_cmd, dict_file)
```

```
This program checks for spellllling misstakes using Unix spell comand
```

```
nalini@nalini-VirtualBox:~$ python3 spell_cmd.py spell_cmd.txt /usr/share/dict/w
ords
Misspelled words:
spellllling
misstakes
Unix
comand
```