National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	Operating Systems	**Section:**	BCS-4A
Program:	BSCS	**Semester:**	Spring 2024
Duration:	1.5 Hours	**Total Marks:**	40
Evaluation Type:	Lab Mid Exam	**Weightage:**	30%
Name:		**Roll Number:**	

Instructions:

- **The quality of the code will affect the marks.**
- Students will receive **ZERO** marks if the answers are plagiarized.
- Use of ANY helping material/code, or cell phones, **INTERNET** and flash drive are strictly prohibited.
- You can use Linux man pages for help.
- You must ensure proper submission of your code following the file naming instructions (given below).
- No queries will be entertained.
- Your submission will contain your **CODE**.
- Submission location:

File Naming Instructions:

- Name your each individual file as **ROLLNUMBER_QUESTION_FILENAME.c** for example **20L_1234_question1_speaker.c**
- **5 marks will be deducted if the naming instructions are not followed.**

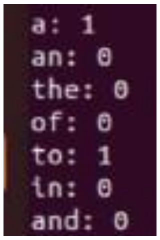
Question 1: (20 Marks)

- Create an application using 3 C files named **create_pipe.c**, **speaker.c**, and **listener.c.**
- **create_pipe.c** create a pipe named **story_pipe.**
- **speaker.c** sends an Alan Turing quote to the **listener.c** via the named pipe created above.
- The listener.c reads this quote from the pipe and count the frequencies of the stop words in it.
- It also **prints** these frequencies on the screen alongside the stopwords.
- Also, write a suitable **makefile** to compile the above files.

Send this quote: "**A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.**"

List of stop words: **[a, the, an, of, to, in, and]**

Output can be:



```
a: 1
an: 0
the: 0
of: 0
to: 1
in: 0
and: 0
```

Note: Strictly follow the naming instructions given at the start of paper.

Question 2: (20 Marks)

- Create five programs named **main.c**, **find_middle.c**, **prime_length.c**, **word_count.c**, and **sort.c**. Also, create a header file named **functions.h** that contains the prototypes of all the functions you will use in the rest of the five programs (e.g., **reverse()**, **find_length()**, etc.).
- The main program (**main.c**) takes a string from the user through command-line arguments or during the program execution. It prints it on the screen along with its PID. Then it executes the **prime_length** program and passes the string to it while calling the **exec** system call.
- The **prime_length** program checks if the length of the string is prime or not and prints it on the screen along with its PID. You cannot use any built-in method to find the length of the string.
- Then it executes the **find_middle** program and passes the string to it while calling the **exec** system call.
- The **find_middle** program finds the middle character of the string and prints it on the screen along with its PID. You cannot use any built-in method to calculate the length, including **strlen** or any other method.
- Then it executes the **word_count** program and passes the string to it while calling the **exec** system call.
- The **word_count** program counts the number of words in the string and prints it on the screen along with its PID.
- Then it executes the **sort** program and passes the string to it via an ordinary pipe.
- The **sort** program sorts the string in ascending order and prints it on the screen along with its PID. You cannot use any built-in method to sort the string.

Note: You can define and use the following functions:

- **int is_prime(int n)**: Checks if the given number **n** is prime.
- **int find_middle_char(char *str)**: Finds the middle character of the given string **str**.
- **void count_vowel_frequencies(char *str)**: Counts the frequencies of vowels in the given string **str**.
- **void sort_string(char *str)**: Sorts the characters of the given string **str** in ascending order.

Note: Strictly follow the naming instructions given at the start of paper.

*********** **GOOD LUCK** ***********