National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Operating Systems**	**Course Code:**	**CS 2006**
Program:	**BSE (5B)**	**Semester:**	**Fall 2024**
Due Date	**15 September, 2024 11:59 PM**	**Total Marks:**	**100 marks**
Type:	**Assignment 1**	**Page(s):**	**3**

Important Instructions:

1. Submit the solution in a folder as your roll number.
2. **You are not allowed to copy solutions from other students. We will check your code for plagiarism using plagiarism checkers. If any sort of cheating is found, heavy penalties will be given to all students involved.**
3. Late submission of your solution is not allowed.

Question 1: Custom Shell [50 marks]

Imagine you've just joined a tech company as a junior developer. Your manager, impressed by your programming skills, assigns you a unique task to streamline the team's daily operations. The team frequently relies on various command-line instructions for their work, but they find switching between different tools a bit cumbersome. They are looking for a more unified, efficient way to handle their command-line needs. Your task is to build a command interpreter program in C that allows them to execute a range of common operations, tailored specifically to their workflow.

Here are the key steps you'll need to implement:

1. When a team member uses your program, they might type an instruction such as cp ./OS ../newOS. Your program should capture this input and store it appropriately (for instance, in a character array or string object).
2. The program will perform tokenization of the input and separate the command and its arguments.
3. It should then create a child process to execute the given instruction, using system calls such as execvp or execlp.
4. Your program must support commands like ls, cp, cd, pwd, mv, mkdir, rmdir, rm, touch, and grep. (You can find more details on these commands https://www.hostinger.com/tutorials/linux-commands .)
5. After executing a command, it should prompt the user to enter another instruction, allowing for continuous interaction.

6. The program should terminate when the user types "exit."

Question 2: Implementing Process Management for an Automated Production Line [25 marks]

You are developing a process management program for a company's automated production line, which consists of four stages: **Material Preparation**, **Assembly**, **Quality Check**, and **Packaging**.

Your task is to create a C program that:

1. **Stage Execution**: Launches a separate process for each stage, in order, where each stage simulates its work by sleeping for a few seconds.
2. **Process Coordination**: Waits for each stage to complete before starting the next one. (Hint: Use waitpid() to wait for the completion of a child process). If a stage fails (simulated with an error code), logs the failure, and attempts to restart it up to two times.
3. **Error Handling**: If a stage continues to fail, logs the issue and terminates the production line.
4. **Completion Report**: Upon successful completion of all stages, displays a summary report with the result and number of attempts for each stage.

Requirements:

- Use appropriate process management techniques to handle creation, synchronization, and error handling. (Hint: Make sure to use fork(), waitpid(), exit(), and signal handling correctly).
- Simulate random failures for each stage (e.g., with a 30% chance).

Question 3: Secure Data Transformation [25 marks]

As a security analyst in an investigation firm, you need to securely transform a critical file, classified_data.txt, that's crucial to an ongoing case. The file is compromised and needs to be either decrypted or redacted to ensure it's safe for analysis.

Your Task:

- **Develop a C/C++ program** that performs secure data transformation using parent-child process interaction.

Details:

1. **File Transformation Options:**
   - **Decrypt Data (Option 1):** Shift each character backward by one (Caesar cipher) if a decryption key is available.
   - **Redact Data (Option 2):** Replace sensitive information, like SSNs, with "[REDACTED]".
2. **Program Flow:**
   - Prompt the user for the input and output file names.
   - Ask the user to choose a transformation option (Decrypt or Redact).
   - Create a child process to perform the selected transformation securely.
   - Ensure that sensitive data is not exposed.
   - The parent process should handle file writing and generate a summary report.

Requirements:

- Handle invalid file names and transformation choices.
- Provide error messages for any issues encountered.

Sample Scenarios:

- **Decrypt Data:** If classified_data.txt contains "Uifsf jt b tfdsfu npofz" and you use a shift of 1, the output should be "There is a secret message" in output.txt.

- **Redact Data:** If classified_data.txt contains “SSN: 123-45-6789”, replace the SSN with “[REDACTED]” in output.txt.

This scenario requires secure and efficient data handling to ensure that the file is correctly transformed and sensitive information remains protected.