**string National University of Computer and Emerging Sciences, Lahore Campus**

|  | **Course:** | **Operating Systems** | **Course Code:** | **CS 2006** |
| --- | --- | --- | --- | --- |
|  | **Program:** | **BSSE (5C)** | **Semester:** | **Fall 2024** |
|  | **Due Date** |  | **Total Marks:** | **75 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.

4. **The use of ofstream/ifstream/fget/fputs etc. is not allowed.**

**Question 1 : Student Record Management System [25 marks]**

You are tasked with developing a student record management system for a small educational institution. The system should manage student data stored in a text file, adhering to specific format requirements. The institution has strict rules for file operations as only open, read, write etc. can be used.

**Requirements:**

Write a C/C++ program that takes two arguments:

1. File Argument: A filename or a path to a file that contains rows of comma-separated data. The second column contains the student ID (Roll Number), and the third column contains the student name. Example format:

1,21L-1234, "Javeria Ali", 3.5

1. Directory Argument: A directory name or a path to a directory where the program may save or read additional files.

**The program should exit with a helpful error message under the following conditions:**

* 1. Argument Count: If the number of arguments is less than or greater than 2.
  2. File Validity: If the file does not exist, is not a file, or is a file of size 0.
  3. Directory Validity: If the directory does not exist or is not a directory.
  4. Format Validation: If any Roll Number does not match the specified format, if any GPA exceeds 4.0, or if there are duplicate Roll Numbers in the file.

**The program should:**

1. Read Student IDs: Read all the student IDs from the file and display the total number of students.
2. Add Data: Allow the user to add new student data during runtime. Ensure that new Roll Numbers follow the required format, and the GPA is within the valid range. Reject duplicate Roll Numbers.
3. Update Data: Enable the user to update existing student data during runtime based on the Roll Number. Ensure the updated data follows all constraints.
4. Delete Data: Permit the user to delete a student’s data during runtime from the file based on the Roll Number.

**Question 2 : Lightweight Shell for Embedded Systems [25 marks]**

You’ve been assigned to develop a custom shell for a new line of embedded devices used in industrial settings. The shell must be lightweight, reliable, and capable of handling standard Unix-like commands. Given the constraints of embedded systems, the shell should be efficient in memory usage and robust in handling user inputs and process management.

**Task:**

Your task is to implement this custom shell with the following key features:

1. Command Input and Execution: The shell should continuously prompt the user to enter a command “cp ./OS ../newOS” storing the input in a dynamically allocated string.
2. Implement tokenization to parse the command and its arguments. Handle edge cases such as extra spaces and special characters.
3. Process Management: Create a child process using fork() and execute the command using execvp. Ensure that the shell waits for the command to complete before prompting the user again.
4. Implement error handling to display meaningful messages if the command fails (e.g., command not found).
5. Loop and Exit: The shell should keep running until the user enters "exit", handling variations like " Exit " or "EXIT".

**Question 3 : Secure Configuration File Management [25 marks]**

As the lead IT administrator for a critical infrastructure facility, you are responsible for managing and securing the system configuration files, specifically system\_config.txt. This file contains sensitive information such as server credentials, network configurations, and access keys, which need to be either encrypted or sanitized before being distributed to different departments.

**Task:**

Develop a C/C++ program that securely manages the configuration file using parent-child process interaction. The program should handle the following options:

1. **Encrypt Configuration (Option 1):** Apply a simple encryption algorithm XOR with a key to hide sensitive data in the file.
2. **Sanitize Configuration (Option 2):** Identify and remove or replace sensitive information, such as passwords and access keys, with **[PRIVATE]** before sharing the file.

**Program Flow:**

1. Prompt the user to provide the input configuration file and the name of the output file.
2. Ask the user to choose a management option (Encrypt or Sanitize).
3. Create a child process to securely execute the chosen file management task.
4. Ensure that sensitive data is not exposed or mishandled during the operation.
5. The parent process should be responsible for saving the processed data into the output file and generating a summary report of the actions performed.

**Requirements:**

1. Handle invalid file names and management options gracefully.
2. Provide clear error messages for any issues encountered.
3. Implement secure processing to ensure that sensitive information remains protected at all stages.

**Hints:**

**Hint 1**: To implement XOR encryption, iterate through each character of the file's content and apply an XOR operation with a repeating key. This can be done using a simple loop.

**Hint 2:** To sanitize the configuration, read each line of the file and search for keywords like "password=" or "access\_key=". When these are found, replace the value with [PRIVATE].

**Hint 3:** Parent- Use fork() to create a child process. The child process should perform the encryption or sanitization. The parent process should wait for the child process to complete using wait(NULL).

**Hint 4:** If an error occurs in the child process (e.g., invalid option), use exit(1) to terminate the child process with an error code.