Q1.

|  |
| --- |
| Write a C program that creates two child processes. The parent process should perform the following tasks:   a)Dynamically allocate memory for a string in the parent process.  Ensure proper memory management to avoid memory leaks.  b)Write the string to a pair of pipes, one for each child process.  c)Create two child processes.  Create two pairs of pipes for communication between the parent and child processes.  Implement error handling mechanisms for pipe creation operations.  Display appropriate error messages if pipe creation fails.  d)Each child process should read from its assigned pipe, modify the string in a unique way, and write the modified string back to another pipe.  Implement two child processes that modify the received string in unique ways.  Each child communicates through a separate pair of pipes.  e)Introduce a synchronization mechanism(e.g., mutex, semaphore) to coordinate the reading and writing operations between the parent and child processes.  f)The parent process should read the modified strings from the child processes and display them. |

Q2.

|  |
| --- |
| Given the Classroom structure that represents a classroom with students. The structure has two members: the number of students n, and an integer array grades containing the students' grades.  struct Classroom {  int n; // Number of students  int grades[1000]; // Array of students' grades  };  Implement the following functions within the Classroom structure:  a. Find the Highest Grade:  Implement a function int maximum() that returns the highest grade in the classroom.  b. Find the Second-Highest Grade:  Implement a function int second\_maximum() that returns the second-highest grade in the classroom.  c. Calculate XOR of Grades:  Implement a function int Find\_Xor() that calculates and returns the XOR of all the grades in the classroom. Threading is allowed for this function.   Constraints:  There are at least 2 and at most 1000 students in the classroom.  Each student's grade is between 1 and 1000. |

Q3.

|  |
| --- |
| Given an alphanumeric string, determine its goodness and calculate its score.  A)Implement a function int Good\_String(char\* str) that checks if a given string is good or not. A string is considered good if it has more vowels (a, e, i, o, u) than consonants.  Input: An alphanumeric string str.  Output: Return 1 if the string is good, otherwise return 0.   Example  char inputStr[] = "abc123";  int goodness = Good\_String(inputStr);  // goodness should be 0   b)Implement a function int Sum\_Of\_Digits(char\* str) that calculates and returns the sum of the digits in a given alphanumeric string.  Input: An alphanumeric string str.  Output: Return the sum of digits in the string.   Example  char inputStr[] = "abc123";  int score = Sum\_Of\_Digits(inputStr);  // score should be 6   c)Modify the Sum\_Of\_Digits function to make use of threading   Constraints:   The length of the input string is between 1 and 10^3.  Each character in the string is either a lowercase letter ('a' to 'z') or a digit ('0' to '9'). |