**Task 5: Description**

The provided program is a **Student Management System** that allows managing a list of students through various operations like adding, removing, searching, and displaying students. The program uses Object-Oriented Programming (OOP) principles, with classes like Student, Student Management System, and the main class Student management system to perform these operations.

**Components of the Program:**

1. **class:**
   * This class represents a student and contains the following private instance variables:
     + name: The student's name.
     + rollnumber: The student's roll number (unique identifier).
     + grade: The student's grade.
   * The class has a constructor to initialize the values of these variables and getter methods (such as getRollNumber) to access the roll number.
   * The toString() method is overridden to provide a human-readable format when displaying the student details (name, roll number, and grade).
2. **Student Management System class:**
   * This class represents the core system that manages a list of students.
   * It contains the following operations:
     + **addStudent(Student student)**: Adds a student to the list of students.
     + **removeStudent(int rollnumber)**: Removes a student based on their roll number. It returns true if the student was removed successfully, otherwise false.
     + **searchStudent(int rollNumber)**: Searches for a student by their roll number and returns the corresponding Student object, or null if the student is not found.
     + **getAllStudents()**: Returns the list of all students in the system.
   * The students list is a dynamic collection (ArrayList) used to store the student objects.
3. **Student\_management\_system class (Main class):**
   * This class is the entry point of the program, where user interaction happens.
   * It uses a Scanner object to take input from the user and allows the following operations through a menu-driven interface:
     + **1. Add Student**: Prompts the user for the student's name, roll number, and grade, then adds the student to the system.
     + **2. Remove Student**: Prompts for the roll number of the student to remove. If found, the student is removed; otherwise, a "not found" message is displayed.
     + **3. Search Student**: Prompts for the roll number of the student to search. If found, the student's details are displayed.
     + **4. Display All Students**: Displays a list of all students in the system. If the list is empty, it notifies the user.
     + **5. Exit**: Exits the program.
4. **Program Flow:**
   * The program uses a while(true) loop to continuously present a menu until the user chooses to exit by selecting option 5.
   * After each operation, the program gives feedback to the user (e.g., confirming the addition, removal, or search results).
   * If an invalid choice is made, it prompts the user to choose again