OBJECT ORIENTED PROGRAMMING :

PLACEMENT MANAGEMENT SYSTEM

CONTENTS

INTRODUCTION

Placement Management System is a project built using the concept of Object Oriented Programming. It helps in easy and efficient management of placement-related activities in any institution. There are currently 3 user levels - Student, Company and College. Each of these users are provided with a variety of features and functionalities for a smooth experience.

It allows users to manage student records and provides features for adding students, displaying student information, searching for students by roll number, and offering a simple user interface.

OBJECTIVES

- **Adding Students**: The code allows users to add student records by providing their name, roll number, and CGPA. This objective enables the system to store information about new students.

- **Displaying Student Information**: Users can view a list of all students in the system. This objective helps in presenting the current status of students registered in the placement system.

- **Searching for Students**: Users can search for a specific student by entering their roll number. This objective facilitates easy retrieval of information about individual students from the system.

- **Simple User Interface**: The code provides a straightforward command-line interface that makes it easy for users to interact with the placement management system. This objective ensures user-friendliness and accessibility.

- **Data Storage**: The code utilizes a vector data structure to store student records. This objective ensures that student data is organized and retrievable for future use.

- **Error Handling**: The code includes basic error handling, such as notifying the user when no students are found or displaying an error message for invalid user inputs. This objective enhances the robustness of the system.

- **Graceful Termination**: The program allows users to exit the system gracefully, ensuring that the user experience is complete and user-friendly.

- **Feedback:** The system provides feedback to the user after certain operations (e.g., adding a student) to confirm the successful execution of tasks. This objective ensures that users receive confirmation of their actions.

- **Code Modularity**: The code is structured using classes to separate concerns (e.g., Student and Placement System). This objective promotes code modularity and maintainability.

- **Documentation:** While not explicitly present in the code, a good practice would be to document the code, including comments, to make it easier for other developers to understand and extend it. Documentation enhances code readability and maintainability.

PROGRAM CODE

```
#include <iostream>
#include <string>

using namespace std;

class Student {
public:
  Student(const string& name, int rollNumber) : name(name),
     rollNumber(rollNumber), isPlaced(false) {}

  string getName() const {
    return name;
  }

  int getRollNumber() const {
    return rollNumber;
  }

  bool isPlacedInCompany() const {
    return isPlaced;
  }
```

```
Void placeInCompany() {
    isPlaced = true;
  }

private:
  string name;
  int rollNumber;
  bool isPlaced;
};

class Company {
public:
  Company(const string& name) : name(name), employeeCount(0) {}

  string getName() const {
    return name;
  }

  int getEmployeeCount() const {
    return employeeCount;
  }
```

```
Void addEmployee() {
      employeeCount++;
   }

private:
   string name;
   int employeeCount;
};

class PlacementManager {
public:
   PlacementManager() : studentCount(0), companyCount(0) {}

   void addStudent(const string& name, int rollNumber) {
      if (studentCount < MAX_STUDENTS) {
         students[studentCount] = new Student(name, rollNumber);
         studentCount++;
      } else {
         cout << "Max student capacity reached!" << endl;
      }
   }
```

```
Void addCompany(const string& name) {
    if (companyCount < MAX_COMPANIES) {
        companies[ companyCount ] = new Company(name);
        companyCount++;
    } else {
        cout << "Max company capacity reached!" << endl;
    }
}

void displayStudents() const {
    cout << "Students:" << endl;
    for (int i = 0; i < studentCount; ++i) {
        cout << "Name: " << students[ i ]->getName() << ", Roll Number: " <<
     students[ i ]->getRollNumber();
        if (students[ i ]->isPlacedInCompany()) {
            cout << " (Placed)" << endl;
        } else {
            cout << " (Not Placed)" << endl;
        }
    }
}
```

```
Void displayCompanies() const {
    cout << "Companies:" << endl;
    for (int i = 0; i < companyCount; ++i) {
        cout << "Name: " << companies[i]->getName() << ", Number of Employees:
    " << companies[i]->getEmployeeCount() << endl;
    }
}

void placeStudent(int studentRollNumber, const string& companyName) {
    for (int i = 0; i < studentCount; ++i) {
        if (students[i]->getRollNumber() == studentRollNumber) {
            students[i]->placeInCompany();
            for (int j = 0; j < companyCount; ++j) {
                if (companies[j]->getName() == companyName) {
                    companies[j]->addEmployee(); // Update the employee count
                    cout << "Student " << students[i]->getName() << " is placed at " <<
    companies[j]->getName() << endl;
                    return;
                }
            }
        }
    }
```

```
        cout << "Student or Company not found." << endl;
    }

    ~PlacementManager() {
        for (int i = 0; i < studentCount; ++i) {
            delete students[i];
        }
        for (int i = 0; i < companyCount; ++i) {
            delete companies[i];
        }
    }

private:
    static const int MAX_STUDENTS = 100;
    static const int MAX_COMPANIES = 100;

    Student* students[MAX_STUDENTS];
    Company* companies[MAX_COMPANIES];
    int studentCount;
    int companyCount;
};
```

```
Int main() {
   PlacementManager manager;
   int choice;
   while (true) {
      cout << "Placement Management System Menu" << endl;
      cout << "1. Add Student" << endl;
      cout << "2. Add Company" << endl;
      cout << "3. Display Students" << endl;
      cout << "4. Display Companies" << endl;
      cout << "5. Place Student" << endl;
      cout << "6. Exit" << endl;
      cout << "Enter your choice: ";
      cin >> choice;

Switch (choice) {
         case 1: {
            string name;
            int rollNumber;
            cout << "Enter student name: ";
            cin.ignore();
            getline(cin, name);
            cout << "Enter roll number: ";
            cin >> rollNumber;
```

```
Manager.addStudent(name, rollNumber);
            break;
        }
        case 2: {
            string name;
            cout << "Enter company name: ";
            cin.ignore();
            getline(cin, name);
            manager.addCompany(name);
            break;
        }
        case 3:
            manager.displayStudents();
            break;
        case 4:
            manager.displayCompanies();
            break;
        case 5: {
            int rollNumber;
            string companyName;
            cout << "Enter student's roll number: ";
            cin >> rollNumber;
```

```
Cout << “Enter company name: “;
            cin.ignore();
            getline(cin, companyName);
            manager.placeStudent(rollNumber, companyName);
            break;
        }
        case 6:
            return 0;
        default:
            cout << “Invalid choice. Please try again.” << endl;
    }
  }

  return 0;
}
```

Output

```
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 1
Enter student name: Jithendar
Enter roll number: 40
```

```
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 1
Enter student name: Nishpreet
Enter roll number: 227
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 1
Enter student name: Tejasvini
Enter roll number: 101
```

```
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 1
Enter student name: Zyaan
Enter roll number: 269
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 1
Enter student name: Ali
Enter roll number: 262
```

```
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 2
Enter company name: Microsoft
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 2
Enter company name: Infosys
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 2
Enter company name: Delloite
```

```
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 5
Enter student's roll number: 40
Enter company name: Microsoft
Student Jithendar is placed at Microsoft
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 5
Enter student's roll number: 227
Enter company name: Delloite
Student Nishpreet is placed at Delloite
```

```
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 5
Enter student's roll number: 101
Enter company name: Microsoft
Student Tejasvini is placed at Microsoft
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 5
Enter student's roll number: 269
Enter company name: Infosys
Student Zyaan is placed at Infosys
```

```
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 5
Enter student's roll number: 262
Enter company name: Infosys
Student Ali is placed at Infosys
Placement Management System Menu
1. Add Student
2. Add Company
3. Display Students
4. Display Companies
5. Place Student
6. Exit
Enter your choice: 3
Students:
Name: Jithendar, Roll Number: 40 (Placed)
Name: Nishpreet, Roll Number: 227 (Placed)
Name: Tejasvini, Roll Number: 101 (Placed)
Name: Zyaan, Roll Number: 269 (Placed)
Name: Ali, Roll Number: 262 (Placed)
```

Classes and Objects in the Code:

1. **Class: Student**

   - The class defines private member variables, such as '**name**', '**rollNumber**', and '**isPlaced**', to store a student's name, roll number, and placement status.
   - It provides public member functions to interact with and retrieve information about a student, including '**getName()', 'getRollNumber()', 'isPlacedInCompany()', and 'placeInCompany()'.**
   - An object of this class represents an individual student and encapsulates their data and behavior.

Classes and Objects in the Code:

2. **Class: Company**

- It defines private member variables, including **name** and **employeeCount**, to store the company's name and the number of employees.
- Public member functions like **getName()** and **getEmployeeCount()** are provided to access company information.
- Instances of this class represent different companies and hold their respective data.

Classes and Objects in the Code:

3. **Class: PlacementManager**

   - It maintains an array of **Student** objects (student records) and an array of **Company** objects (company records).
   - The class provides methods like **addStudent(), addCompany(), displayStudents(), displayCompanies(),** and **placeStudent().**
   - Objects of this class manage and manipulate student and company data within the system.

WORKING/EXPLANATION

The C++ source code implements a simple Student Placement Management System. It allows users to interact with the system through a text-based menu-driven interface. The code includes basic error handling to notify the user in case of maximum capacity reached for students or companies or when a student or company is not found. Here's how the code works:

- Initialization: The code starts by defining and initializing the Student and Company classes to represent students and companies. It also declares a Placement Manager class, which acts as the central entity for managing student and company records.

- User Interaction Loop: The program enters an infinite loop that presents a menu to
  the user. The user is prompted to select an option from the menu.

WORKING/EXPLANATION

The menu options are as follows:

1. Add a new student.
2. Add a new company.
3. Display a list of students and their job status.
4. Display a list of companies and the number of employees.
5. Place a student in a company.
6. Exit the program when you're done

CONCLUSION

This C++ code for a Placement Management System provides a practical tool for organizing student and company data. Its user-friendly interface simplifies data management, making it valuable for educational institutions and job placement services. The usage of classes and objects and error management by this code helps in data handling, ensuring efficient and user-friendly operation.

THANK YOU