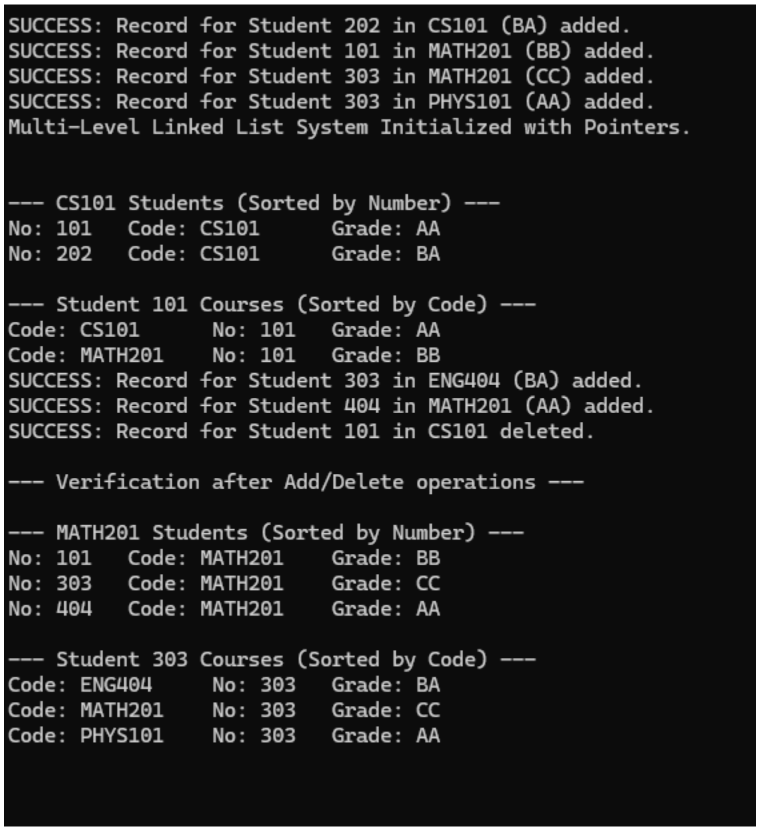
İSİM : NATHANAELLE BOPTI NGAH BONG
ÖĞRENCİ NUMERASI: 24080410150
Bilgisayar Mühendisliği

LİNKED LİST 2. ÖDEV

```
using System;
using System.Collections.Generic;
using System.Linq;
public class StudentCourseNode
{
    public int StudentNumber;
    public string CourseCode;
    public string LetterGrade;


    public StudentCourseNode NextCourseForStudent  = null;

    public StudentCourseNode NextStudentInCourse { get; set; } = null;
}

public class SchoolManagementLinkedList
{
    private Dictionary<int, StudentCourseNode> StudentHeads = new Dictionary<int,
StudentCourseNode>();
    private Dictionary<string, StudentCourseNode> CourseHeads { get; set; } = new
Dictionary<string, StudentCourseNode>();
    private List<StudentCourseNode> AllNodes { get; set; } = new
List<StudentCourseNode>();

    public SchoolManagementLinkedList()
    {
        AddNode(101, "CS101", "AA");
        AddNode(202, "CS101", "BA");
        AddNode(101, "MATH201", "BB");
        AddNode(303, "MATH201", "CC");
        AddNode(303, "PHYS101", "AA");
    }
    public void AddNode(int studentNo, string courseCode, string grade)
    {
        if (AllNodes.Any(n => n.StudentNumber == studentNo && n.CourseCode ==
courseCode))
        {
            Console.WriteLine($"ERROR: Student {studentNo} is already registered for
course {courseCode}.");
            return;
        }

        var newNode = new StudentCourseNode
        {
            StudentNumber = studentNo,
            CourseCode = courseCode,
            LetterGrade = grade
        };
        if (StudentHeads.ContainsKey(studentNo))
        {
            var current = StudentHeads[studentNo];
            while (current.NextCourseForStudent != null)
            {
                current = current.NextCourseForStudent;
            }
            current.NextCourseForStudent = newNode;
        }
        else
        {
            StudentHeads[studentNo] = newNode;
        }

        if (CourseHeads.ContainsKey(courseCode))
        {
            var current = CourseHeads[courseCode];
            while (current.NextStudentInCourse != null)
            {
                current = current.NextStudentInCourse;
            }
            current.NextStudentInCourse = newNode;
        }
        else
        {
            CourseHeads[courseCode] = newNode;
        }
        AllNodes.Add(newNode);
        Console.WriteLine($"SUCCESS: Record for Student {studentNo} in {courseCode}
({grade}) added.");
    }
    public void DeleteNode(int studentNo, string courseCode)
    {
        var targetNode = AllNodes.FirstOrDefault(n => n.StudentNumber == studentNo
&& n.CourseCode == courseCode);

        if (targetNode == null)
        {
            Console.WriteLine($"ERROR: Record for Student {studentNo} in
{courseCode} not found.");
            return;
        }
        if (StudentHeads.ContainsKey(studentNo))
        {
            StudentCourseNode studentPrev = null;
            var current = StudentHeads[studentNo];

            while (current != null && (current.StudentNumber != studentNo ||
current.CourseCode != courseCode))
            {
                studentPrev = current;
                current = current.NextCourseForStudent;
            }

            if (studentPrev == null)
            {
                StudentHeads[studentNo] = targetNode.NextCourseForStudent;
            }
            else
            {
                studentPrev.NextCourseForStudent = targetNode.NextCourseForStudent;
            }

            // If the head is null, remove the head entry for cleanup
            if (StudentHeads[studentNo] == null)
            {
                StudentHeads.Remove(studentNo);
            }
        }

        if (CourseHeads.ContainsKey(courseCode))
        {
            StudentCourseNode coursePrev = null;
            var current = CourseHeads[courseCode];

            while (current != null && (current.StudentNumber != studentNo ||
current.CourseCode != courseCode))
            {
                coursePrev = current;
                current = current.NextStudentInCourse;
            }

            if (coursePrev == null)
            {
                CourseHeads[courseCode] = targetNode.NextStudentInCourse;
            }
            else
            {
                coursePrev.NextStudentInCourse = targetNode.NextStudentInCourse;
            }
            if (CourseHeads[courseCode] == null)
            {
                CourseHeads.Remove(courseCode);
            }
        }

        // Remove from the global list
        AllNodes.Remove(targetNode);
        Console.WriteLine($"SUCCESS: Record for Student {studentNo} in {courseCode}
deleted.");
    }
    public void ListStudentsInCourse(string courseCode)
    {
        Console.WriteLine($"\n--- {courseCode} Students (Sorted by Number) ---");

        if (!CourseHeads.ContainsKey(courseCode))
        {
            Console.WriteLine($"'{courseCode}' course has no registered students.");
            return;
        }

        // Traversal
        var foundNodes = new List<StudentCourseNode>();
        var current = CourseHeads[courseCode];
        while (current != null)
        {
            foundNodes.Add(current);
            current = current.NextStudentInCourse;
        }
        var sortedNodes = foundNodes.OrderBy(n => n.StudentNumber).ToList();

        foreach (var node in sortedNodes)
        {
            Console.WriteLine($"No: {node.StudentNumber,-5} Code: {node.CourseCode,-
10} Grade: {node.LetterGrade}");
        }
    }
    public void ListCoursesByStudent(int studentNo)
    {
        Console.WriteLine($"\n--- Student {studentNo} Courses (Sorted by Code) ---
");

        if (!StudentHeads.ContainsKey(studentNo))
        {
            Console.WriteLine($"Student {studentNo} is not registered for any
courses.");
            return;
        }
        var foundNodes = new List<StudentCourseNode>();
        var current = StudentHeads[studentNo];
        while (current != null)
        {
            foundNodes.Add(current);
            current = current.NextCourseForStudent;
        }
        var sortedNodes = foundNodes.OrderBy(n => n.CourseCode).ToList();

        foreach (var node in sortedNodes)
        {
            Console.WriteLine($"Code: {node.CourseCode,-10} No:
{node.StudentNumber,-5} Grade: {node.LetterGrade}");
        }
    }
}

public class Program
{
    public static void Main(string[] args)
    {
        var system = new SchoolManagementLinkedList();
        Console.WriteLine("Multi-Level Linked List System Initialized with
Pointers.\n");
        system.ListStudentsInCourse("CS101");
        system.ListCoursesByStudent(101);
        system.AddNode(303, "ENG404", "BA");
        system.AddNode(404, "MATH201", "AA");
        system.DeleteNode(101, "CS101");
        Console.WriteLine("\n--- Verification after Add/Delete operations ---");
        system.ListStudentsInCourse("MATH201");
        system.ListCoursesByStudent(303);

        Console.ReadKey();
    }
}
```



```
SUCCESS: Record for Student 202 in CS101 (BA) added.
SUCCESS: Record for Student 101 in MATH201 (BB) added.
SUCCESS: Record for Student 303 in MATH201 (CC) added.
SUCCESS: Record for Student 303 in PHYS101 (AA) added.
Multi-Level Linked List System Initialized with Pointers.


--- CS101 Students (Sorted by Number) ---
No: 101   Code: CS101      Grade: AA
No: 202   Code: CS101      Grade: BA

--- Student 101 Courses (Sorted by Code) ---
Code: CS101      No: 101   Grade: AA
Code: MATH201    No: 101   Grade: BB
SUCCESS: Record for Student 303 in ENG404 (BA) added.
SUCCESS: Record for Student 404 in MATH201 (AA) added.
SUCCESS: Record for Student 101 in CS101 deleted.

--- Verification after Add/Delete operations ---

--- MATH201 Students (Sorted by Number) ---
No: 101   Code: MATH201    Grade: BB
No: 303   Code: MATH201    Grade: CC
No: 404   Code: MATH201    Grade: AA

--- Student 303 Courses (Sorted by Code) ---
Code: ENG404     No: 303   Grade: BA
Code: MATH201    No: 303   Grade: CC
Code: PHYS101    No: 303   Grade: AA
```