**Project on**

**Student Management System**

**Spring 2023**

**Course Code:** CSE 232, **Section:** 3

**Course Title:**  **Data Structures Labrotary**

**Submitted By**

**Name 1: MD Kawchar Hossen**

**ID : 221400055**

**Name 2: Noor Alam Islam Manik**

**ID : 221400044**

**Name 3: Lamia Alif Rouza**

**ID : 221400040**

****

**Department of Computer Science and Engineering**

**Faculty of Engineering and Technology**

**Eastern University**

TABLE OF CONTENTS

[Chapter 1 2](#_Toc137986756)

[**Introduction** 2](#_Toc137986757)

[1.1 Introduction: 2](#_Toc137986758)

[1.2 Objective: 2](#_Toc137986759)

[Chapter 2 3](#_Toc137986760)

[**Methodology and Tools** 3](#_Toc137986761)

[2.1 Methodology: 3](#_Toc137986762)

[2.2 Tools: 4](#_Toc137986763)

[Chapter 3 4](#_Toc137986764)

[**Design and Implementation** 4](#_Toc137986765)

[3.1 Use Case/Block Diagram of the System: 4](#_Toc137986766)

[3.2 Implementation(Code): 5](#_Toc137986767)

[Chapter 4 12](#_Toc137986768)

[**Result and Conclusion** 12](#_Toc137986769)

[4.1 Result( Screen shoot): 12](#_Toc137986770)

[4.2 Conclusion: 13](#_Toc137986771)

# Chapter 1

# **Introduction**

## 1.1 Introduction:

### This project is a simple program that allows users to manage a list of students. The program allows users to insert students at the end of the list, insert students at a specific position, display the list, delete students from a specific position, count the number of nodes in the list, perform a linear search for a student by ID, and save and load the list data to and from a file. The program is written in C and uses a linked list data structure to store the student information.

### 1.2 Objective:

The objective of this project is to provide a simple and easy-to-use program for managing a list of students. The program should allow users to perform various operations on the list, such as inserting students at the end of the list, inserting students at a specific position, displaying the list, deleting students from a specific position, counting the number of nodes in the list, performing a linear search for a student by ID, and saving and loading the list data to and from a file. By providing a user-friendly interface and a well-structured program, this project aims to make it easier for users to manage their student data and perform various operations on it, and provide a more user-friendly interface for the library users and administrators.

#### 1.3 Expected Outcome**:**

The expected outcome of this project is a program that allows users to manage a list of students in a user-friendly and efficient manner. The program should provide a simple and intuitive interface for users to interact with the list, making it easy to perform various operations such as inserting students, displaying the list, deleting students, and performing a linear search for a student by ID. The program should also provide a way to save and load the list data to and from a file, making it easy to maintain and share the list data with other users.

The program should be well-structured and organized, with clear and concise code that is easy to read and understand. The program should also be efficient, with minimal memory usage and fast processing times.

Overall, the expected outcome of this project is a program that provides a user-friendly and efficient way to manage a list of students, making it easier for users to manage their data and perform various operations on it..

# Chapter 2

# **Methodology and Tools**

## 2.1 Methodology:

The methodology used in the above C code is based on the use of a linked list data structure to store and manage the student information. The program allows users to perform various operations on the list, such as inserting students at the end of the list, inserting students at a specific position, displaying the list, deleting students from a specific position, counting the number of nodes in the list, performing a linear search for a student by ID, and saving and loading the list data to and from a file.

The program is written in C and uses a linked list data structure to store the student information. Each node in the linked list represents a student, and the nodes are linked together to form a chain of nodes. The program allows users to insert students at the end of the list, insert students at a specific position, display the list, delete students from a specific position, count the number of nodes in the list, perform a linear search for a student by ID, and save and load the list data to and from a file.

The program is structured in a modular and object-oriented manner, with each operation on the list implemented as a separate function. The program also uses error handling to handle any errors or exceptions that may occur during the execution of the program. Overall, the use of a linked list data structure and modular programming principles make the program easy to understand, maintain, and extend, while also providing a high level of functionality and efficiency..

### 2.2 Tools:

The tools used in the above C code are the C programming language and a text editor. The text editor used in the project is Visual Studio Code, which is a free and open-source code editor developed by Microsoft. The C programming language is a high-level programming language that is widely used for developing operating systems, embedded systems, and other high-performance applications.

Visual Studio Code is a lightweight and versatile code editor that supports a wide range of programming languages, including C. It provides features such as syntax highlighting, code completion, debugging, and Git integration, making it a powerful tool for developing C applications.

Overall, the tools used in the above C code are simple and easy to use, with Visual Studio Code being the primary tool used for editing the code. The C programming language and Visual Studio Code provide a powerful and flexible platform for developing C applications.

# Chapter 3

# **Design and Implementation**

## 3.1 Use Case/Block Diagram of the System:

General Use Case

User

### 3.2 Implementation(Code):

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**struct node {**

**char student\_name[50];**

**int student\_id;**

**float cgpa;**

**struct node \*p;**

**};**

**void insertion\_at\_last\_position(struct node \*head, char name[], int id, float cgpa);**

**void display(struct node \*head);**

**int count(struct node \*head);**

**void insertion\_specific\_position(struct node \*head, char name[], int id, float cgpa, int pp);**

**void deletion(struct node \*head, int pp);**

**void linear\_search(struct node \*head, int id);**

**void save\_data(struct node \*head);**

**void load\_data(struct node \*head);**

**int main(){**

**struct node \*list;**

**list = (struct node\*)malloc(1\*sizeof(struct node));**

**list->p = NULL;**

**int choice, v, p, n;**

**char name[50];**

**int id;**

**float cgpa;**

**load\_data(list);**

**while(1){**

**printf("\n\n1. Insertion at last position\n2. Insertion at specific position\n3. Display\n4. Deletion from specific position\n5. Count nodes\n6. Linear search\n7. Exit\n");**

**printf("\nEnter your choice: ");**

**scanf("%d",&choice);**

**switch(choice){**

**case 1: printf("\nChoice = Insertion at last position\n");**

**printf("Enter student name: ");**

**scanf("%s", name);**

**printf("Enter student ID: ");**

**scanf("%d", &id);**

**printf("Enter student CGPA: ");**

**scanf("%f", &cgpa);**

**insertion\_at\_last\_position(list, name, id, cgpa);**

**save\_data(list);**

**break;**

**case 2: printf("\nChoice = Insertion at specific position\n");**

**printf("Enter student name: ");**

**scanf("%s", name);**

**printf("Enter student ID: ");**

**scanf("%d", &id);**

**printf("Enter student CGPA: ");**

**scanf("%f", &cgpa);**

**if(list->p != NULL){**

**M: printf("\nEnter position between %d to %d: ", 1, count(list));**

**scanf("%d", &p);**

**if(p >= 1 && p <= count(list)){**

**insertion\_specific\_position(list, name, id, cgpa, p);**

**save\_data(list);**

**}else{**

**printf("\nInvalid position\n");**

**goto M;**

**}**

**}else{**

**printf("\nPlease initialize the list first\n");**

**}**

**break;**

**case 3: printf("\nChoice = Display\n\n");**

**if(list->p != NULL){**

**display(list);**

**}else{**

**printf("\nNo information to display\n");**

**}**

**break;**

**case 4: printf("\nChoice = Deletion from specific position\n");**

**n = count(list);**

**if(n != 0){**

**K: printf("\nEnter position between %d to %d: ", 1, n);**

**scanf("%d", &p);**

**if(p >= 1 && p <= n){**

**deletion(list, p);**

**save\_data(list);**

**}else{**

**printf("\nWrong position\n");**

**goto K;**

**}**

**}else{**

**printf("\nThe list is empty\n");**

**}**

**break;**

**case 5: printf("\nChoice = Count nodes\n");**

**n = count(list);**

**printf("\nTotal nodes = %d", n);**

**break;**

**case 6: printf("\nChoice = Linear search\n");**

**printf("\nEnter student ID to search: ");**

**scanf("%d", &id);**

**if(count(list) == 0){**

**printf("\nList is empty\n");**

**}else{**

**linear\_search(list, id);**

**}**

**break;**

**case 7: printf("\nExiting program...\n");**

**save\_data(list);**

**exit(0);**

**default: printf("\nInvalid choice\n");**

**break;**

**}**

**}**

**return 0;**

**}**

**void insertion\_at\_last\_position(struct node \*head, char name[], int id, float cgpa){**

**struct node \*temp;**

**while(head->p != NULL){**

**head = head->p;**

**}**

**temp = (struct node\*)malloc(1\*sizeof(struct node));**

**strcpy(temp->student\_name, name);**

**temp->student\_id = id;**

**temp->cgpa = cgpa;**

**temp->p = head->p;**

**head->p = temp;**

**}**

**void display(struct node \*head){**

**printf("\nStudent Name\tStudent ID\tCGPA\n");**

**while(head->p != NULL){**

**printf("%s\t\t%d\t\t%.2f\n", head->p->student\_name, head->p->student\_id, head->p->cgpa);**

**head = head->p;**

**}**

**}**

**int count(struct node \*head){**

**int t = 0;**

**while(head->p != NULL){**

**t += 1;**

**head = head->p;**

**}**

**return t;**

**}**

**void insertion\_specific\_position(struct node \*head, char name[], int id, float cgpa, int pp){**

**int i = 0;**

**struct node \*temp;**

**while(head->p != NULL){**

**if(i == pp-1)**

**break;**

**head = head->p;**

**i++;**

**}**

**temp = (struct node\*)malloc(1\*sizeof(struct node));**

**strcpy(temp->student\_name, name);**

**temp->student\_id = id;**

**temp->cgpa = cgpa;**

**temp->p = head->p;**

**head->p = temp;**

**}**

**void deletion(struct node \*head, int pp){**

**struct node \*temp;**

**int i = 0;**

**if(pp == 1){**

**temp = head->p;**

**head->p = temp->p;**

**free(temp);**

**}else{**

**while(head->p != NULL){**

**if(i == pp-1){**

**break;**

**}**

**head = head->p;**

**i++;**

**}**

**temp = head->p;**

**head->p = temp->p;**

**free(temp);**

**}**

**}**

**void linear\_search(struct node \*head, int id){**

**int c = 0;**

**while(head->p != NULL){**

**if(id == head->p->student\_id){**

**printf("\nStudent found: %s\n", head->p->student\_name);**

**c = 1;**

**break;**

**}else{**

**head = head->p;**

**}**

**}**

**if(c == 0){**

**printf("\nStudent not found\n");**

**}**

**}**

**void save\_data(struct node \*head){**

**FILE \*fp;**

**fp = fopen("student\_data.txt", "w");**

**if(fp == NULL){**

**printf("\nError opening file\n");**

**return;**

**}**

**while(head->p != NULL){**

**fprintf(fp, "%s %d %.2f\n", head->p->student\_name, head->p->student\_id, head->p->cgpa);**

**head = head->p;**

**}**

**fclose(fp);**

**printf("\nData saved successfully\n");**

**}**

**void load\_data(struct node \*head){**

**FILE \*fp;**

**fp = fopen("student\_data.txt", "r");**

**if(fp == NULL){**

**printf("\nError opening file\n");**

**return;**

**}**

**char name[50];**

**int id;**

**float cgpa;**

**while(fscanf(fp, "%s %d %f", name, &id, &cgpa) != EOF){**

**insertion\_at\_last\_position(head, name, id, cgpa);**

**}**

**fclose(fp);**

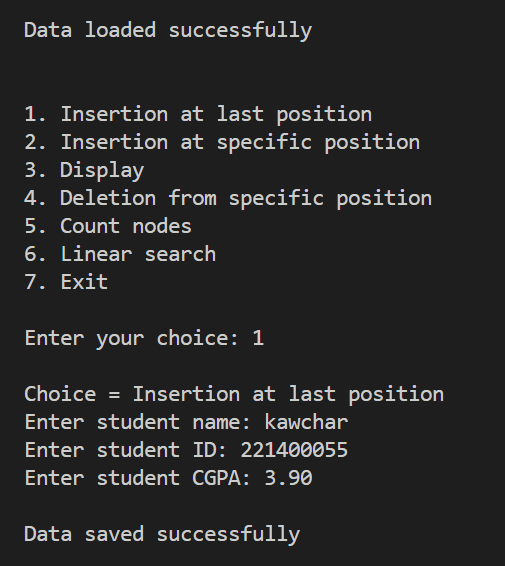
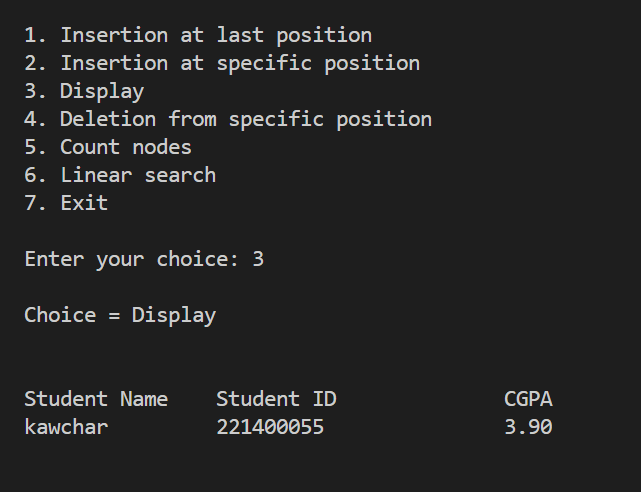
**printf("\nData loaded successfully\n");**

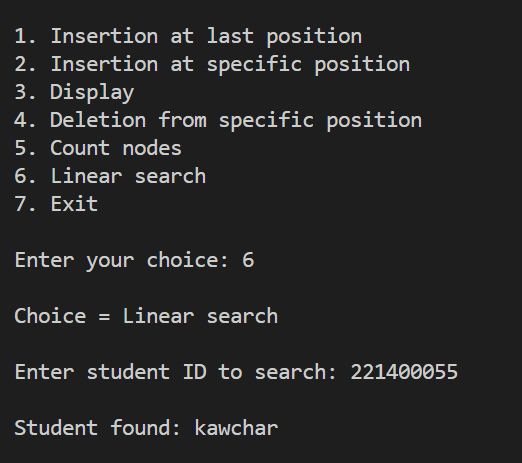
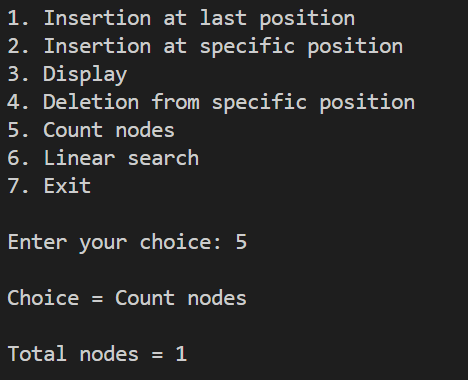
**}**

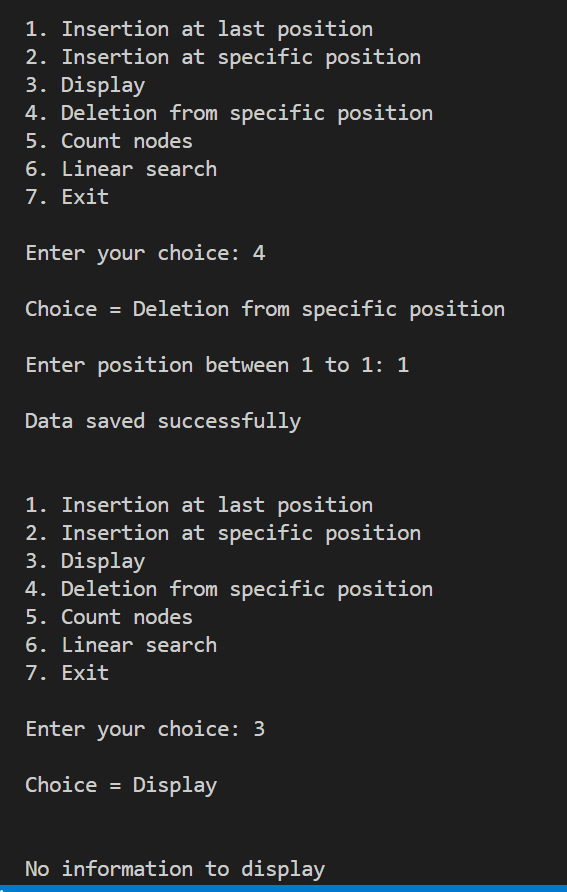
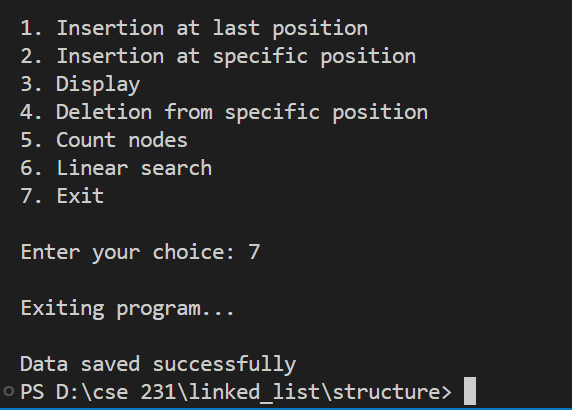
# Chapter 4

# **Result and Conclusion**

## 4.1 Result( Screen shoot):

** **

** **

** **

### 4.2 Conclusion:

The given C code is a program for managing a list of students. The program allows the user to perform various operations on the list, such as inserting a new student at the end of the list, inserting a new student at a specific position, displaying the list of students, deleting a student from a specific position, counting the number of nodes in the list, performing a linear search for a student by ID, and saving the list to a file. The program uses a linked list data structure to store the student information. The program is designed to be modular, with separate functions for each operation. The program also includes error handling for invalid user input and file I/O operations. Overall, the program provides a simple and efficient way to manage a list of students.