**Task 1 in c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#define MAX\_STUDENTS 100

#define MAX\_GRADE\_LEN 3

// Global array to store student grades

char grades[MAX\_STUDENTS][MAX\_GRADE\_LEN];

int count = 0;

// Function prototypes

void addGrade();

void deleteGrade();

void searchGrade();

void calculateAverage();

void findMinMax();

void displayAll();

void displayMenu();

int main() {

displayMenu();

return 0;

}

void addGrade() {

if (count >= MAX\_STUDENTS) {

printf("Error: Maximum number of students reached!\n");

return;

}

printf("Enter student grade (A-F with optional +/-): ");

scanf("%s", grades[count]);

// Validate grade format

if (strlen(grades[count]) > MAX\_GRADE\_LEN ||

!isalpha(grades[count][0]) ||

(strlen(grades[count]) > 1 && !strchr("+-", grades[count][1])) ||

(strlen(grades[count]) > 2 && grades[count][1] != '\0')) {

printf("Invalid grade format! Use A-F with optional +/-\n");

return;

}

// Convert to uppercase

grades[count][0] = toupper(grades[count][0]);

count++;

printf("Grade added successfully!\n");

}

// delete elements in the array

void deleteGrade() {

if (count == 0) {

printf("No grades to delete!\n");

return;

}

char search[MAX\_GRADE\_LEN];

printf("Enter grade to delete: ");

scanf("%s", search);

search[0] = toupper(search[0]);

int found = 0;

for (int i = 0; i < count; i++) {

if (strcmp(grades[i], search) == 0) {

// Shift remaining elements

for (int j = i; j < count - 1; j++) {

strcpy(grades[j], grades[j+1]);

}

count--;

found = 1;

printf("Grade deleted successfully!\n");

break;

}

}

if (!found) {

printf("Grade not found!\n");

}

}

//searching in the array

void searchGrade() {

if (count == 0) {

printf("No grades to search!\n");

return;

}

char search[MAX\_GRADE\_LEN];

printf("Enter grade to search: ");

scanf("%s", search);

search[0] = toupper(search[0]);

int found = 0;

for (int i = 0; i < count; i++) {

if (strcmp(grades[i], search) == 0) {

printf("Grade found at position %d\n", i+1);

found = 1;

}

}

if (!found) {

printf("Grade not found!\n");

}

}

// finding the average of grades

void calculateAverage() {

if (count == 0) {

printf("No grades to calculate average!\n");

return;

}

float sum = 0;

for (int i = 0; i < count; i++) {

switch(grades[i][0]) {

case 'A': sum += 4; break;

case 'B': sum += 3; break;

case 'C': sum += 2; break;

case 'D': sum += 1; break;

case 'F': sum += 0; break;

}

}

printf("Average grade: %.2f\n", sum / count);

}

void findMinMax() {

if (count == 0) {

printf("No grades to analyze!\n");

return;

}

char min[MAX\_GRADE\_LEN], max[MAX\_GRADE\_LEN];

strcpy(min, grades[0]);

strcpy(max, grades[0]);

for (int i = 1; i < count; i++) {

if (strcmp(grades[i], min) < 0) {

strcpy(min, grades[i]);

}

if (strcmp(grades[i], max) > 0) {

strcpy(max, grades[i]);

}

}

printf("Highest grade: %s\n", max);

printf("Lowest grade: %s\n", min);

}

void displayAll() {

if (count == 0) {

printf("No grades to display!\n");

return;

}

printf("Current grades:\n");

for (int i = 0; i < count; i++) {

printf("%d: %s\n", i+1, grades[i]);

}

}

// display menu for easy navigation

void displayMenu() {

int choice;

do {

printf("\nStudent Grade Management System\n");

printf("1. Add a grade\n");

printf("2. Delete a grade\n");

printf("3. Search for a grade\n");

printf("4. Calculate average grade\n");

printf("5. Find highest and lowest grades\n");

printf("6. Display all grades\n");

printf("0. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1: addGrade(); break;

case 2: deleteGrade(); break;

case 3: searchGrade(); break;

case 4: calculateAverage(); break;

case 5: findMinMax(); break;

case 6: displayAll(); break;

case 0: printf("Exiting...\n"); break;

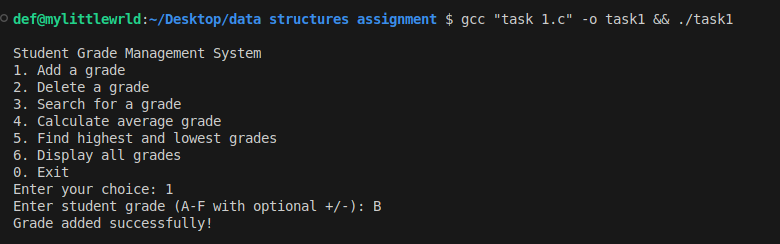
default: printf("Invalid choice!\n");

}

} while (choice != 0);

}

**Task 1 sample outputs:**

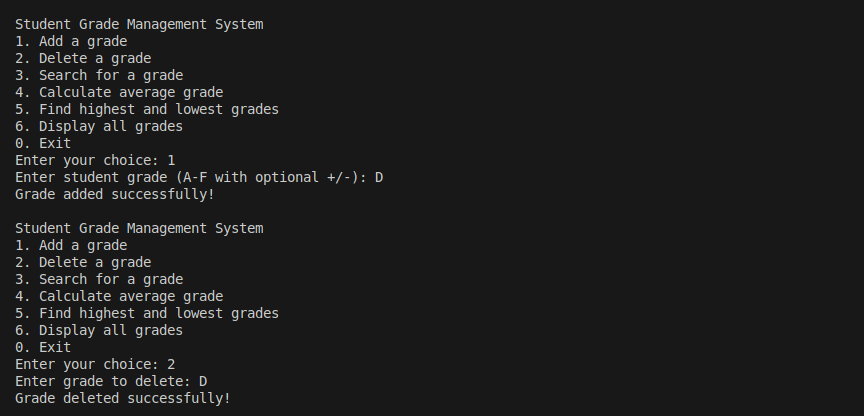
****

**Fig 1.0 adding a grade**

**Calculating the average grade

**

**Fig 1.1 calculating the average grade**

****

**Fig 1.2 adding and deleting**

****

**Fig 1.3 Search for a grade**

**Task 2**

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

#define MAX\_STUDENTS 25

#define MAX\_NAME\_LEN 50

typedef struct {

char name[MAX\_NAME\_LEN];

char grade;

} Student;

// Function prototypes

void bubbleSort(Student arr[], int n, bool sortByName);

void selectionSort(Student arr[], int n, bool sortByName);

void insertionSort(Student arr[], int n, bool sortByName);

void printStudents(Student arr[], int n);

void displayMenu();

// Comparison functions

int compareByName(const Student \*a, const Student \*b) {

return strcmp(a->name, b->name);

}

int compareByGrade(const Student \*a, const Student \*b) {

return a->grade - b->grade;

}

void inputStudents(Student students[], int \*count) {

printf("Enter number of students (max %d): ", MAX\_STUDENTS);

scanf("%d", count);

getchar(); // Consume newline

for (int i = 0; i < \*count; i++) {

printf("\nStudent %d:\n", i+1);

printf("Name: ");

fgets(students[i].name, MAX\_NAME\_LEN, stdin);

students[i].name[strcspn(students[i].name, "\n")] = '\0'; // Remove newline

printf("Grade (A-F): ");

scanf(" %c", &students[i].grade);

getchar(); // Consume newline

// Validate grade

students[i].grade = toupper(students[i].grade);

while (students[i].grade < 'A' || students[i].grade > 'F') {

printf("Invalid grade! Enter A-F: ");

scanf(" %c", &students[i].grade);

students[i].grade = toupper(students[i].grade);

getchar();

}

}

}

int main() {

Student students[MAX\_STUDENTS];

int count = 0;

inputStudents(students, &count);

displayMenu(students, count);

return 0;

}

void bubbleSort(Student arr[], int n, bool sortByName) {

for (int i = 0; i < n-1; i++) {

for (int j = 0; j < n-i-1; j++) {

int compareResult;

if (sortByName) {

compareResult = compareByName(&arr[j], &arr[j+1]);

} else {

compareResult = compareByGrade(&arr[j], &arr[j+1]);

}

if (compareResult > 0) {

// Swap

Student temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

void selectionSort(Student arr[], int n, bool sortByName) {

for (int i = 0; i < n-1; i++) {

int min\_idx = i;

for (int j = i+1; j < n; j++) {

int compareResult;

if (sortByName) {

compareResult = compareByName(&arr[j], &arr[min\_idx]);

} else {

compareResult = compareByGrade(&arr[j], &arr[min\_idx]);

}

if (compareResult < 0) {

min\_idx = j;

}

}

// Swap

Student temp = arr[min\_idx];

arr[min\_idx] = arr[i];

arr[i] = temp;

}

}

void insertionSort(Student arr[], int n, bool sortByName) {

for (int i = 1; i < n; i++) {

Student key = arr[i];

int j = i - 1;

while (j >= 0) {

int compareResult;

if (sortByName) {

compareResult = compareByName(&arr[j], &key);

} else {

compareResult = compareByGrade(&arr[j], &key);

}

if (compareResult > 0) {

arr[j + 1] = arr[j];

j--;

} else {

break;

}

}

arr[j + 1] = key;

}

}

void printStudents(Student arr[], int n) {

printf("\nStudent Records:\n");

printf("----------------\n");

for (int i = 0; i < n; i++) {

printf("%-15s %c\n", arr[i].name, arr[i].grade);

}

printf("\n");

}

void displayMenu(Student students[], int count) {

int choice;

bool sortByName;

Student copy[MAX\_STUDENTS];

do {

printf("\nStudent Record Sorting System\n");

printf("1. Bubble Sort\n");

printf("2. Selection Sort\n");

printf("3. Insertion Sort\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

if (choice >= 1 && choice <= 3) {

printf("\nSort by:\n");

printf("1. Name\n");

printf("2. Grade\n");

printf("Enter your choice: ");

int sortChoice;

scanf("%d", &sortChoice);

sortByName = (sortChoice == 1);

// Make a copy of original array for sorting

memcpy(copy, students, sizeof(Student) \* count);

switch(choice) {

case 1:

bubbleSort(copy, count, sortByName);

printf("\nSorted using Bubble Sort:\n");

break;

case 2:

selectionSort(copy, count, sortByName);

printf("\nSorted using Selection Sort:\n");

break;

case 3:

insertionSort(copy, count, sortByName);

printf("\nSorted using Insertion Sort:\n");

break;

}

printStudents(copy, count);

} else if (choice != 4) {

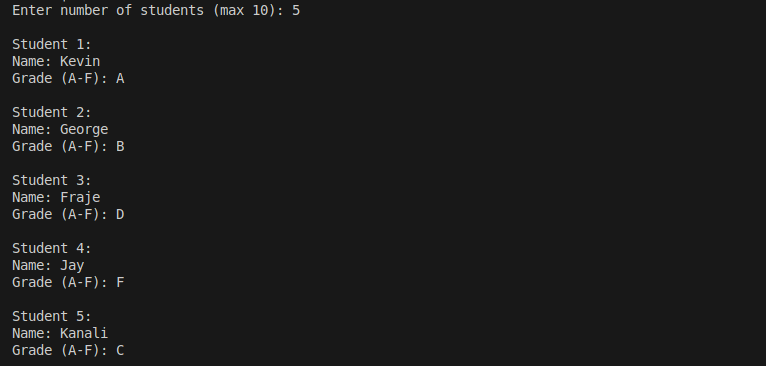
printf("Invalid choice!\n");

}

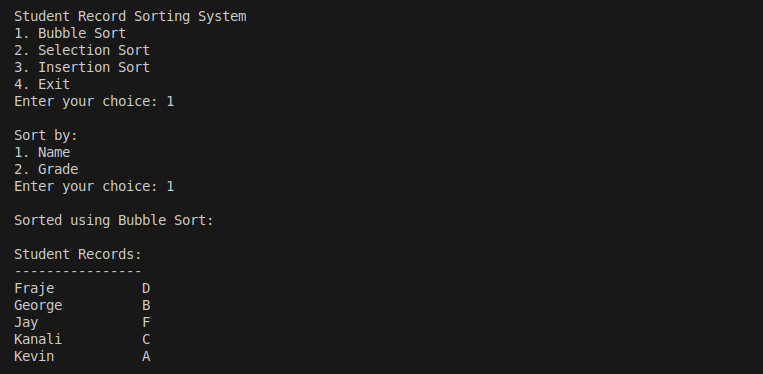
} while (choice != 4);

}

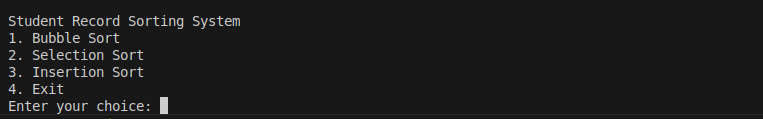
**Sample output snippets**

****

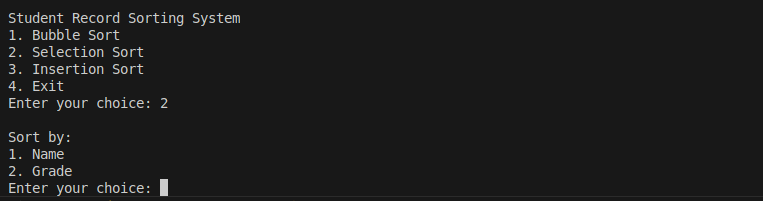
**FIG 2.0 Adding the records**

****

**FIG 2.1 sorting the records with bubble sort using names**

****

**Fig 2.2 shows the options for sorting in the programm**

****

**Fig 2.3 sorting can be done through names or grades in each sorting alogarithm**

**Task 3**

#include <iostream>

using namespace std;

// (i) Define Node class

class Node {

public:

int Data; // Integer data

Node\* Next; // Pointer to next node

};

int main() {

// (ii) Create node instance

Node node1;

// (iii) Declare and assign head pointer

Node\* head = &node1;

// (iv) Assign integer value

node1.Data = 42; // Example value

// Verify the implementation

cout << "Node data: " << head->Data << endl;

cout << "Next pointer: " << head->Next << endl;

return 0;

}

**Sample output snippets**

****

Submission:

Capture the source code and sample outputs for each of the exercises into ONE ms-word document and follow this link to upload your word document by 14th April 2025.

Link: <https://forms.gle/G1oEZB4G279pEKTo6>