

# STUDENT FEEDBACK TRACKING SYSTEM

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
#include <string.h>
```

```
typedef struct node {
```

```
    int studentID;
```

```
    char courseCode[20];
```

```
    int rating;
```

```
    char comments[200]
```

```
    struct node *next;
```

```
} Node;
```

```
Node * addFeedback (Node * head, int id, const char * course, int rating,  
                    const char * comments);
```

```
void searchByStudent (Node * head, int studentID);
```

```
void searchByCourse (Node * head, const char * courseCode);
```

```
void searchByCourse (Node * head, const char * courseCode);
```

```
void displayReverse (Node * head);
```

```
Node * cloneList (Node * head);
```

```
void displayList (Node * head);
```

```
void freeList (Node * head);
```

```
int main () {
```

```
    Node * feedbackList = NULL;
```

```
    feedbackList = addFeedback (feedbackList, 101, "CS101", 5,
```

```
                                "Excellent teaching");
```

```
    feedbackList = addFeedback (feedbackList, 102, "CS101", 4, "Good but
```

```
                                could improve);
```

```
    feedbackList = addFeedback (feedbackList, 103, "CS201", 3, "Average");
```

```
    feedbackList = addFeedback (feedbackList, 104, "CS101", 2, "Too much theory);
```

```
    feedbackList = addFeedback (feedbackList, 105, "PH301", 4, "Interesting  
                                lectures");
```

```

printf ("%ln-- All feedbacks --ln");
displayList (FeedbackList);
printf ("%ln-- Search by Student ID 102 --ln");
searchBy Student (FeedbackList, 102);
printf ("%ln-- Search by Course CS101 --ln");
searchBy Course (FeedbackList, '101');

printf ("%ln-- Average for CS101 --ln");
averageBy Course (feedback, "CS101");
printf ("%ln- feedbacks in Reverse Order --ln");
displayReverse (FeedbackList);

printf ("%ln-- Cloning FeedbackList --ln");
Node *clone = cloneList (FeedbackList);
displayList (clone);
freeList (FeedbackList);
freeList (clone);
return 0;
}

```

3

// functions

```

Node *addFeedback (Node *head, int id, const char *
course, int rating, const char *
comments)
{
Node *newNode = (Node *) malloc (sizeof (Node));
if (!newNode) {
printf ("Memory allocation failed ln");
return head;
}
newNode->StudentID = id;
strcpy (NewNode->Course Code, course);
newNode->rating = rating;
}

```

```
strcpy (newNode->comments, comments);
```

```
newNode->next = NULL;
```

```
if (!head) return newNode; // first node
```

```
Node *temp = head;
```

```
while (temp->next) temp = temp->next;
```

```
temp->next = newNode;
```

```
return head;
```

```
}
```

```
void displayList (Node *head) {
```

```
Node *temp = head;
```

```
while (temp) {
```

```
printf ("StudentID : %d | course : %s | Rating : %d | Comment :  
%s\n",
```

```
temp->StudentID, temp->courseCode, temp->  
rating, temp->comments);
```

```
temp = temp->next;
```

```
}
```

```
void searchByStudent (Node *head, int studentID) {
```

```
Node *temp = head;
```

```
int found = 0;
```

```
while (temp) {
```

```
• if (temp->studentID == studentID) {
```

```
printf ("Found: %d | %s | Rating : %d | comment : %s\n",
```

```
temp->studentID, temp->coursecode, temp->rating,  
temp->comments);
```

```
found = 1
```

```
}
```

```
temp = temp->next;
```

```
}
```

```
if (!found) printf ("No feedback found for student %d\n",  
studentID);
```

```
}
```

// search feedback by Course Code

```
void searchByCourse (Node *head, const char *coursecode) {
```

```
    Node *temp = head;
```

```
    int found = 0;
```

```
    while (temp) {
```

```
        if (strcmp(temp->coursecode, coursecode) == 0) {
```

```
            printf ("found: Student %d, Rating: %d, Comment: %s\n",
```

```
                temp->studentID, temp->rating, temp->comments);
```

```
            found = 1;
```

```
        }
```

```
        temp = temp->next;
```

```
    }
```

```
    if (!found) printf ("No feedback found for course %s\n", coursecode);
```

```
}
```

// Calculate average rating for a course

```
void averageByCourse (Node *head, const char *coursecode) {
```

```
    Node *temp = head;
```

```
    int count = 0, sum = 0;
```

```
    while (temp) {
```

```
        if (strcmp(temp->coursecode, coursecode) == 0) {
```

```
            sum += temp->rating;
```

```
            count ++;
```

```
        }
```

```
        temp = temp->next;
```

```
    }
```

```
    if (count == 0)
```

```
        printf ("No feedback for course %s\n", coursecode);
```

```
    else
```

```
        printf ("Average rating for %s = %2f\n", coursecode,
```

```
                (float) sum / count);
```

```
}
```



// Display list in reverse Order (recursive)

void displayReverse(Node \*head) {

if (!head) return;

displayReverse(head->next);

printf("Student ID: %d | Course: %s | Rating: %.1d | Comment: %s\n",

head->StudentID, head->CourseCode, head->rating,

head->Comments);

}

// Clone the entire feedback list

Node \*cloneList(Node \*head) {

if (!head) return NULL;

Node \*cloneHead = NULL, \*cloneTail = NULL;

Node \*temp = head;

while (temp) {

Node \*newNode = (Node \*) malloc(sizeof(Node));

\*newNode = \*temp

newNode->next = NULL;

if (!cloneHead) {

cloneHead = cloneTail = newNode; }

else {

cloneTail->next = newNode;

cloneTail = newNode;

}

temp = temp->next;

}

return cloneHead;

}

// free memory

void freeList (Node \*head)

```
{  
    Node *temp;  
    while (head) {  
        temp = head;  
        head = head->next;  
        free(temp);  
    }  
}
```

main.c

Run

Output

Clear

```

105 }
106
107 // Search feedback by Course Code
108 void searchByCourse(Node *head, const char
    *courseCode) {
109     Node *temp = head;
110     int found = 0;
111     while(temp) {
112         if(strcmp(temp->courseCode, courseCode
            ) == 0) {
113             printf("Found: Student %d, Rating:
                %d, Comment: %s\n",
114                 temp->studentID, temp->rating,
115                 temp->comments);
116             found = 1;
117             temp = temp->next;
118         }
119         if(!found) printf("No feedback found for
            course %s\n", courseCode);
120     }
121
122 // Calculate average rating for a course
123 void averageByCourse(Node *head, const char
    *courseCode) {
124     Node *temp = head;
125     int count = 0, sum = 0;
126     while(temp) {
127         if(strcmp(temp->courseCode, courseCode
            ) == 0) {
128             sum += temp->rating;
129             count++;
130         }
131         temp = temp->next;
132     }
133     if(count == 0)
134         printf("No feedback for course %s\n",
            courseCode);
135     else
136         printf("Average rating for %s = %.
            2f\n", courseCode, (float)sum
            /count);
137 }
138
139 // Display list in reverse order (recursive)
140 void displayReverse(Node *head) {
141     if(!head) return;
142     displayReverse(head->next);
143     printf("StudentID: %d | Course: %s |
        Rating: %d | Comment: %s\n",
144         head->studentID, head->courseCode,
145         head->rating, head->comments);
146 }
147 // Clone the entire feedback list
148 Node* cloneList(Node *head) {
149     if(!head) return NULL;
150     Node *cloneHead = NULL, *cloneTail = NULL;
151     Node *temp = head;
152     while(temp) {
153         Node *newNode = (Node*)malloc(sizeof
            (Node));
154         *newNode = *temp; // Copy all fields
155         newNode->next = NULL;
156
157         if(!cloneHead) {
158             cloneHead = cloneTail = newNode;
159         } else {
160             cloneTail->next = newNode;
161             cloneTail = newNode;
162         }
163         temp = temp->next;
164     }
165     return cloneHead;
166 }
167
168 // Free memory
169 void freeList(Node *head) {
170     Node *temp;
171     while(head) {

```

--- All Feedbacks ---

StudentID: 101 | Course: CS101 | Rating: 5 | Comment:

Excellent teaching

StudentID: 102 | Course: CS101 | Rating: 4 | Comment:

Good but could improve slides

StudentID: 103 | Course: MA201 | Rating: 3 | Comment:

Average class

StudentID: 104 | Course: CS101 | Rating: 2 | Comment:

Too much theory

StudentID: 105 | Course: PH301 | Rating: 4 | Comment:

Interesting lectures

--- Search by Student ID 102 ---

Found: 102, CS101, Rating: 4, Comment: Good but could  
improve slides

--- Search by Course CS101 ---

Found: Student 101, Rating: 5, Comment: Excellent  
teachingFound: Student 102, Rating: 4, Comment: Good but  
could improve slidesFound: Student 104, Rating: 2, Comment: Too much  
theory

--- Average for CS101 ---

Average rating for CS101 = 3.67

--- Feedbacks in Reverse Order ---

StudentID: 105 | Course: PH301 | Rating: 4 | Comment:  
Interesting lecturesStudentID: 104 | Course: CS101 | Rating: 2 | Comment:  
Too much theoryStudentID: 103 | Course: MA201 | Rating: 3 | Comment:  
Average classStudentID: 102 | Course: CS101 | Rating: 4 | Comment:  
Good but could improve slidesStudentID: 101 | Course: CS101 | Rating: 5 | Comment:  
Excellent teaching

--- Cloning Feedback List ---

StudentID: 101 | Course: CS101 | Rating: 5 | Comment:  
Excellent teachingStudentID: 102 | Course: CS101 | Rating: 4 | Comment:  
Good but could improve slidesStudentID: 103 | Course: MA201 | Rating: 3 | Comment:  
Average classStudentID: 104 | Course: CS101 | Rating: 2 | Comment:  
Too much theoryStudentID: 105 | Course: PH301 | Rating: 4 | Comment:  
Interesting lectures

=== Code Execution Successful ===