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 * add Feebback(Node* head, int id, const char*course, int rating,
                                    const char * comments);
void searchByStudent (Node *head, int studentID);
void searchBy Course (Node * head, const char* course code);
void Search By Course ( Node * head, const Char * CourseCode);
Void display Reverse (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 = add feedback (feedbacklist, 104, "CS101", 2, "Toomuch theory");
    feedbacklist = add feedback (feedbacklist, 105, "PH301", 4, "Intresting
                                                    lectures");
```

```
Printf ("\n-- All feedbacks --\n");
displayList (FeedbackList);
printf ("\n--Search by Student ID 102--\n");
SearchByStudent(feedbackList, 102);
printf ("\n-- Search by Course CS101--\n");
SearchBy Course (feedbackList, "101");

Printf ("\n--Average for CS101--\n");
averageBy Course (feedback, "CS101");
Printf( "\n- feedbacks in Revese Order--\n");
displayReverse (feedbackList);
Printf ("\n--Cloning Feedback List--\n");
Node*clone = clonelist (feedbacklist);
displayList(clone);
freeList (feedbackList);
freeList (clone);
return 0;
}
// functions
Node *add feebback (Node *head, int id ,const char*
                        Course, int rating, const char*
                                        comments)
{
  Node *newNode= (Node*) malloc (sizeof (Node));
  if(!newNode){
     printf ("Memory allocation failed\n");
     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",
                                        Student ID);
}
```

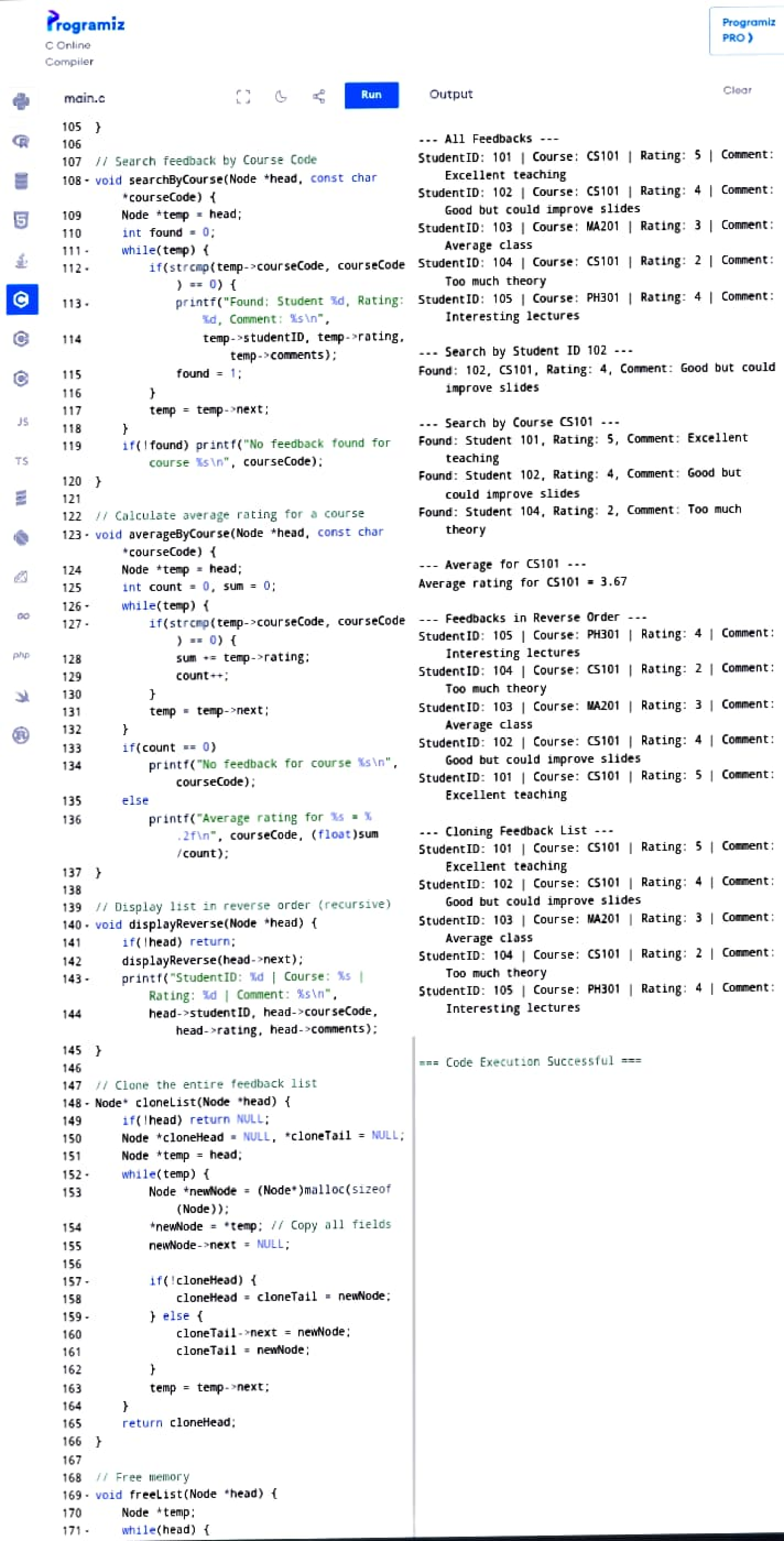
```
// search feedback by Course Code
Void searchByCourse(Node *head, const char *coursecode){
 Node *temp = head;
 int found = 0;
 while (temp){
  if (strcmp(temp->courseCode, coursecode) == 0){
   printf("found: Student %d , Rating: %d, comment: %s\n",
          temp->studentID, temp->rating , temp->comments);
   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"), Couse code,
                                        (float) sum/count);
}
```

```
// Display list in reverse Order (recursive)
void displayReverse(Node * head) {
  if (!head) return;
  displayReverse (head → next);
  printf ("StudentID: %d | course : %s | Rating : %d | 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, *clone Tail = NULL;
Node *temp = head;
while (temp) {
   Node *newNode = (Node*) malloc (sizeof(Node));
   *newNode = *temp
   newNode → next = NULL;
   if ( ! cloneHead) {
     cloneHead = clonetail = NewNode; }
   else {
     clone~~Tail~~ → Next = newNode;
     clone Tail = newnode;
   }
   temp = temp → next;
 }
 return cloneHead;
}
```

```
//free memory

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



```c
}

// 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("StudentID: %d | Course: %s | Rating: %d | 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; // Copy all fields
        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) {
```

```
--- 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 teaching
Found: Student 102, Rating: 4, Comment: Good but could improve slides
Found: 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 lectures
StudentID: 104 | Course: CS101 | Rating: 2 | Comment: Too much theory
StudentID: 103 | Course: MA201 | Rating: 3 | Comment: Average class
StudentID: 102 | Course: CS101 | Rating: 4 | Comment: Good but could improve slides
StudentID: 101 | Course: CS101 | Rating: 5 | Comment: Excellent teaching

--- Cloning Feedback List ---
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


=== Code Execution Successful ===
```