```c
/* BY SUBMITTING THIS FILE TO CARMEN, I CERTIFY THAT I HAVE PERFORMED ALL OF
THE WORK TO CREATE THIS FILE AND/OR DETERMINE THE ANSWERS FOUND WITHIN
THIS FILE MYSELF WITH NO ASSISSTANCE FROM ANY PERSON (OTHER THAN THE
INSTRUCTOR OR GRADERS OF THIS COURSE) AND I HAVE STRICTLY ADHERED TO THE
TENURES OF THE OHIO STATE UNIVERSITY'S ACADEMIC INTEGRITY POLICY.
*/
#define FALSE 0
#define TRUE 1
#include <stdio.h>
#include <stdlib.h>
struct Cat_Grade{
        float score1;
        float score2;
        float score3;
        float cumulative;
};
struct Data {
        char student_name[40];
        int student_ID;
        struct Cat_Grade cat1;
        struct Cat_Grade cat2;
        struct Cat_Grade cat3;
        struct Cat_Grade cat4;
        float current_grade;
        float final_grade;
};
typedef struct Node{
        struct Data student;
        struct Node *next;
} Node;
/*Function descriptions are listed below*/
Node* allocate_node(struct Data statistics); /*malloc() a node and populate it with
student data*/
void deleteNode(Node *head, int id); /*detete a node from the linked list*/
Node* insertNode(Node *list_head, Node *newNodePtr); /*insert a node in the linked
list*/
/*reads in the student data from the disk*/
Node* read_student_data(char *name, int id, float a1, float a2, float a3, float b1,
float b2, float b3, float c1, float c2, float c3, float d1, float d2, float d3);
Node *get_NodeforID(Node *head, int StudentID); /*return a node for the corresponding
ID*/
/*The following functions process the options. For example, option1 processes option1
and etc.*/
void option1(Node *head, char *categories);
void option2(Node *head, char *categories);
void option3(Node *head, char *categories);
void option4(Node *head, char *categories);
void option5(Node *head, char *categories);
void option6(Node *head, char *categories);
void option7(Node *head, char *categories);
void option8(Node* head, char *categories);
void option9(Node* head);
void option10(FILE *file, Node *head, char (*category_names)[15]);
void printStudent(Node* nodePtr); /*Prints a single student to one line*/
void printHeader(char *Category_Names); /*Prints the header before any student*/
int ID_isduplicate(Node *head, int newStudentID); /*Check to see if ID number is a
duplicate*/
void printLine(Node *head, char *Category_Names); /*Prints a line of student data*/
Node* get_NodeforName(Node *head, char *name); /*return a node for the corresponding
name*/
void process_options(Node *list_head, char (*category_names)[15], FILE *file); /
*Process the options listed above*/
```

```
void print_student_data(FILE *file, Node *list_head, char (*category_names)[15]); /
*write all data to disk*/
```