Structure and Dynamic Memory Management assignments

Mandatory

1. Refer the question 1 solved in “Structure and function”. Extend the above program to read a number of records from the user as a single command line argument (each record is delimited by a semicolon and record fields are delimited by comma) and store in an array of structures.

   Sample input and output are given below.

   Input: “user1,90;user21,100, userABC,56,userX,40”;

   Output:

   No. of records: 4

   Record 1:

   Name:user1, Percentage:90

   Record 2:

   Name:user21, Percentage:100

   Record 3:

   Name:userABC, Percentage:56

   Record 4:

   Name:userX, Percentage:40

   Implement all required functions and call them to get the desired output.

   Check for memory leak.

user72@trainux01: ~/Assignments

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_NAME_LEN 100
struct student {
    char name[MAX_NAME_LEN];
    int percentage;
};
int read_records(char *input, struct student **students) {
    int count = 0;
    char *record = strtok(input, ";");
    while (record != NULL) {
        struct student new_student;
        char *name = strtok(record, ",");
        char *percentage = strtok(NULL, ",");
        if (name && percentage) {
            strncpy(new_student.name, name, MAX_NAME_LEN);
            new_student.percentage = atoi(percentage);
            students[count] = (struct student *)malloc(sizeof(struct student));
            if (students[count] == NULL) {
                printf("Memory allocation failed!\n");
                return -1;
            }
            *students[count] = new_student;
            count++;
        }
        record = strtok(NULL, ";");
    }
    return count;
}
void display_records(struct student **students, int count) {
    printf("No. of records: %d\n", count);
    for (int i = 0; i < count; i++) {
        printf("Record %d:\n", i + 1);
        printf("Name: %s, Percentage: %d\n", students[i]->name, students[i]->percentage);
    }
}
char* search_update(char *searchstr, char *replacestr, struct student **students, int count) {
    for (int i = 0; i < count; i++) {
        if (strcmp(students[i]->name, searchstr) == 0) {
            strncpy(students[i]->name, replacestr, MAX_NAME_LEN);
            return students[i]->name;
        }
    }
    return NULL;
}
int delete_record(char *searchstr, int percent, struct student **students, int *count) {
    for (int i = 0; i < *count; i++) {
```

user72@trainux01: ~/Assignments

```
        if (strcmp(students[i]->name, searchstr) == 0 || students[i]->percentage == percent) {
            free(students[i]);
            for (int j = i; j < *count - 1; j++) {
                students[j] = students[j + 1];
            }
            (*count)--;
            return 1;
        }
    }
    return 0;
}
int copy(char *name, struct student **newstudent, struct student **students, int count) {
    for (int i = 0; i < count; i++) {
        if (strcmp(students[i]->name, name) == 0) {
            *newstudent = (struct student *)malloc(sizeof(struct student));
            if (*newstudent == NULL) {
                printf("Memory allocation failed!\n");
                return 0;
            }
            **newstudent = *students[i];
            return 1;
        }
    }
    return 0;
}
void free_records(struct student **students, int count) {
    for (int i = 0; i < count; i++) {
        free(students[i]);
    }
}
int main(int argc, char *argv[]) {
    if (argc != 2) {
        printf("Please provide the records as a command line argument.\n");
        return -1;
    }
    struct student *students[100];
    int count = read_records(argv[1], students);
    if (count < 0) {
        return -1;
    }
    display_records(students, count);
    char *updated_name = search_update("user1", "user100", students, count);
    if (updated_name) {
        printf("Updated name: %s\n", updated_name);
    }
    if (delete_record("userABC", -1, students, &count)) {
        printf("Record deleted successfully.\n");
    }
```

```
    }
    struct student *new_student;
    if (copy("user21", &new_student, students, count)) {
        printf("Copied record: Name: %s, Percentage: %d\n", new_student->name, new_student->percentage);
        free(new_student);  // Free the copied record
    }
    display_records(students, count);
    free_records(students, count);
    return 0;
}
```

```
user72@trainux01:~/Assignments$ ./a.out user1,90;
No. of records: 1
Record 1:
Name: user1, Percentage: 90
Updated name: user100
No. of records: 1
Record 1:
Name: user100, Percentage: 90
```