```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAX_LINE_LENGTH 500
#define MAX_CATEGORY_LENGTH 50

//Samuel did this part-----------------------------------------------------------------
// Define the structure
typedef struct {
    int callNumber;
    char category[MAX_CATEGORY_LENGTH];
    char description[MAX_LINE_LENGTH];
} CallToAction;


//Sehajdeep did this part-------------------------------------------------------------
// Function Prototypes
void displayAllCalls(CallToAction *calls, int totalCalls);
void searchByCategory(CallToAction *calls, int totalCalls, const char *category);
int countTotalCalls(CallToAction *calls, int totalCalls);
void saveByCategory(CallToAction *calls, int totalCalls, const char *category);
char *strToLower(const char *str);


//Samuel did this part-----------------------------------------------------------------
void displayAllCalls(CallToAction *calls, int totalCalls) {
    printf("\nDisplaying all Calls to Action:\n");
    for (int i = 0; i < totalCalls; i++) {
        printf("%d | %s | %s\n", calls[i].callNumber, calls[i].category, calls[i].description);
    }
}

//Sehajdeep did this part-------------------------------------------------------------
void searchByCategory(CallToAction *calls, int totalCalls, const char *category) {
    char *lowerCategory = strToLower(category);
    int found = 0;

    printf("\nCalls to Action in category '%s':\n", category);
    for (int i = 0; i < totalCalls; i++) {
        char *currentCategory = strToLower(calls[i].category);
        if (strcmp(lowerCategory, currentCategory) == 0) {
            printf("%d | %s | %s\n", calls[i].callNumber, calls[i].category, calls[i].description);
            found = 1;
        }
        free(currentCategory);
    }
    free(lowerCategory);

    if (!found) {
        printf("No Calls to Action found in the category '%s'.\n", category);
    }
}

int countTotalCalls(CallToAction *calls, int totalCalls) {
    return totalCalls;
}

//Samuel did this part-----------------------------------------------------------------
void saveByCategory(CallToAction *calls, int totalCalls, const char *category) {
    char filename[100];
    sprintf(filename, "%s_calls.txt", category);
    FILE *file = fopen(filename, "w");
    if (file == NULL) {
        printf("Error: Could not create file '%s'.\n", filename);
        return;
    }

    char *lowerCategory = strToLower(category);
    int found = 0;

    for (int i = 0; i < totalCalls; i++) {
        char *currentCategory = strToLower(calls[i].category);
        if (strcmp(lowerCategory, currentCategory) == 0) {
            fprintf(file, "%d | %s | %s\n", calls[i].callNumber, calls[i].category, calls[i].description);
            found = 1;
        }
        free(currentCategory);
    }
    free(lowerCategory);
    fclose(file);

    if (found) {
        printf("Calls to Action in category '%s' saved to '%s'.\n", category, filename);
    } else {
```

```c
        printf("No Calls to Action found in the category '%s'.\n", category);
        remove(filename);
    }
}

//Sehajdeep did this part-----------------------------------------------------------------
char *strToLower(const char *str) {
    char *lowerStr = malloc(strlen(str) + 1);
    if (lowerStr == NULL) {
        printf("Error: Memory allocation failed.\n");
        exit(1);
    }
    for (int i = 0; str[i]; i++) {
        lowerStr[i] = tolower(str[i]);
    }
    lowerStr[strlen(str)] = '\0';
    return lowerStr;
}

// Main Function
int main() {
    FILE *file = fopen("calls_to_action.txt", "r");
    if (file == NULL) {
        printf("Error: Could not open file.\n");
        return 1;
    }

    // Allocate memory for calls to action
    CallToAction *calls = malloc(sizeof(CallToAction) * 100);
    if (calls == NULL) {
        printf("Error: Memory allocation failed.\n");
        fclose(file);
        return 1;
    }

    int totalCalls = 0;
    char line[MAX_LINE_LENGTH];

    // Read data line by line
    while (fgets(line, sizeof(line), file)) {
        CallToAction *current = &calls[totalCalls];
        sscanf(line, "%d|%[^|]|%[^\n]", &current->callNumber, current->category, current->description);
        totalCalls++;
    }
    fclose(file);

    int choice;
    char category[MAX_CATEGORY_LENGTH];

    // Menu
    do {
        printf("\nMenu:\n");
        printf("1. Display all Calls to Action\n");
        printf("2. Search Calls to Action by category\n");
        printf("3. Display total number of Calls to Action\n");
        printf("4. Save Calls to Action by category to a new file\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        if (choice == 1) {
            displayAllCalls(calls, totalCalls);
        } else if (choice == 2) {
            printf("Enter category (e.g., Education): ");
            scanf(" %[^\n]s", category);
            searchByCategory(calls, totalCalls, category);
        } else if (choice == 3) {
            printf("Total number of Calls to Action: %d\n", countTotalCalls(calls, totalCalls));
        } else if (choice == 4) {
            printf("Enter category to save (e.g., Education): ");
            scanf(" %[^\n]s", category);
            saveByCategory(calls, totalCalls, category);
        } else if (choice == 5) {
            printf("Exiting the program.\n");
        } else {
            printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 5);

    // Free allocated memory
    free(calls);

    return 0;
}
```