```c
/*
Name: Larry Nguyen
Lab #7
Date : 03/5/2020
Description: This program creates an phonebook that holds contact information. New features added.
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// Global variables
int Count = 0;
int DelCount = 0;

// Phonebook list structure
typedef struct PhoneBookList
{
    char *ContactFirstName;
    char *ContactLastName;
    char *ContactPhoneNumber;
} list;

// Phonebook delete function structure
typedef struct DeleteEntry
{
        char *ContactFirstName;
    char *ContactLastName;
} take;

// Structure pointers
list *PhoneMemory;
take *DeletePhoneMemory;

// Prototypes
void Add();
void Delete();
void Display();
void Sort();
void Search();
void Random();
void Reset();

int main(void)
{
        int PhoneBookSelection;
    do {  // Main menu display
        printf("\n\nPhone Book:\n\n");
            printf("1) Add friend\n");
            printf("2) Delete friend\n");
            printf("3) Display phone book\n");
            printf("4) Alphabetically sort phonebook by first name\n");
        printf("5) Find a phone number for a given name\n");
        printf("6) Randomly select contact\n");
        printf("7) Reset Phonebook\n");
            printf("8) Exit\n");
            printf("What do you want to do: ");
            scanf("%d", &PhoneBookSelection);

        // Setting up Switch
        switch (PhoneBookSelection)
                {
                case 1:  // Add a contact
                    Add();
                    break;
                case 2:  // Delete a contact
                Delete();
                break;
            case 3:  // Display phonebook list
                Display();
                break;
                        case 4:  // Alphabetically sort phonebook
                Sort();
                break;
                        case 5:  // Search phonebook using a name
                                Search();
                                break;
                        case 6:  // Randomly select a contact
                                Random();
                                break;
                        case 7:  // Reset entire phonebook
                                Reset();
                                break;
            case 8:  // Break loop and ends the program
                break;
```

```
        defauPhoneMemory:  // Invalid number selection
            printf("\nInvalid selection. Try again.\n");
            break;
        } // End Switch
          } while (PhoneBookSelection != 8);  //End Do While loop

    // Freeing up Memory
    free(DeletePhoneMemory);
    free(PhoneMemory);
    DeletePhoneMemory = NULL;
    PhoneMemory = NULL;
    return 0;
}
//Add an entry
void Add()
{
    if (Count == 0)
     {
        PhoneMemory = (list *) malloc ((Count*25) + 25);
     }
     else
     {
        PhoneMemory = (list *) realloc (PhoneMemory, (Count*50) + 50);
     }
    if (PhoneMemory == NULL)
    {
        printf("Error, no more memory\n");
    }
    else
    {
                // Memory allocation
           PhoneMemory[Count].ContactFirstName = (char *) malloc(sizeof(char)*15);
           PhoneMemory[Count].ContactLastName = (char *) malloc(sizeof(char)*15);
           PhoneMemory[Count].ContactPhoneNumber = (char *) malloc(sizeof(char)*15);
               // Input contact info
                  printf("\nEnter their First Name: ");
           scanf("%s", PhoneMemory[Count].ContactFirstName);
           printf("\nEnter their Last Name: ");
           scanf("%s", PhoneMemory[Count].ContactLastName);
           printf("\nEnter their Phone Number: ");
           scanf("%s", PhoneMemory[Count].ContactPhoneNumber);
           printf("\nContact added\n");
    }
    Count++;
}
//Delete an entry
void Delete()
{
     int i;
     int q = 0;
     char *userName;
    // Memory allocation
     if (DelCount == 0)
     {
            DeletePhoneMemory = (take *) malloc ((DelCount*25) + 25);
     }
     else
     {
        DeletePhoneMemory = (take *) realloc (DeletePhoneMemory, (DelCount*1) + 1);
     }
     if (DeletePhoneMemory == NULL)
     {
        printf("This cannot be deleted (out of memory)\n");
     }
     else
     {
                DeletePhoneMemory[DelCount].ContactFirstName = (char *) malloc(sizeof(char)*15);
        DeletePhoneMemory[DelCount].ContactLastName = (char *) malloc(sizeof(char)*15);
    // User input for deleting contact
                printf("\nEnter Contact's First Name: ");
        scanf("%s", DeletePhoneMemory[DelCount].ContactFirstName);
        printf("\nEnter Contact's Last Name: ");
        scanf("%s", DeletePhoneMemory[DelCount].ContactLastName);
     }
     for (i = 0; i < Count; i++)
     {
        if (PhoneMemory[i].ContactFirstName == NULL && PhoneMemory[i].ContactLastName == NULL) continue;
        if (strcmp(PhoneMemory[i].ContactFirstName, DeletePhoneMemory[DelCount].ContactFirstName) == 0 && strcmp(PhoneMemory[i].
ContactLastName, DeletePhoneMemory[DelCount].ContactLastName) == 0)
        {
          printf("\n%s %s has been deleted\n", PhoneMemory[i].ContactFirstName, PhoneMemory[i].ContactLastName);
          PhoneMemory[i].ContactFirstName = NULL;
          PhoneMemory[i].ContactLastName = NULL;
          PhoneMemory[i].ContactPhoneNumber = NULL;
          q = 1;
          break;
        }
```

```
     } // End for loop

     if (q != 1)
     {
     printf("\nThat contact does not exist\n");
     }
     DelCount++;
     Count--;
}
//Display all phonebook entries
void Display()
{
     int i;
     printf("\nYour contacts:\n");
     for (i = 0; i < Count; i++)
     {
         if (PhoneMemory[i].ContactFirstName != NULL && PhoneMemory[i].ContactLastName != NULL)
         {
            printf("\n%s %s: %s\n", PhoneMemory[i].ContactFirstName, PhoneMemory[i].ContactLastName, PhoneMemory[i].ContactPhoneNumber);
         }
     }// End for loop
     system("pause");
}
void Sort() // Alphabetically sorts by first name
{
     int i;
     int j;
     char temp[75][75];
     printf("\nPhonebook Contacts:\n");
     for (i = 0; i < Count; i++)
     {
             for (j = i + 1; j < Count; j++)
         {
            if (strcmp(PhoneMemory[i].ContactFirstName, PhoneMemory[j].ContactFirstName) > 0)
             {
             strcpy(temp[i], PhoneMemory[i].ContactFirstName); // Moves first name
             strcpy(PhoneMemory[i].ContactFirstName, PhoneMemory[j].ContactFirstName);
             strcpy(PhoneMemory[j].ContactFirstName, temp[i]);
             strcpy(temp[i], PhoneMemory[i].ContactLastName); // Moves last name alongside first name
             strcpy(PhoneMemory[i].ContactLastName, PhoneMemory[j].ContactLastName);
             strcpy(PhoneMemory[j].ContactLastName, temp[i]);
             strcpy(temp[i], PhoneMemory[i].ContactPhoneNumber); // Moves phone number alongside first name
             strcpy(PhoneMemory[i].ContactPhoneNumber, PhoneMemory[j].ContactPhoneNumber);
             strcpy(PhoneMemory[j].ContactPhoneNumber, temp[i]);
             }
         }
     }//End for loop

     for (i = 0; i < Count; i++) // Prints out sorted phonebook
     printf("\n%s %s %s\n", PhoneMemory[i].ContactFirstName, PhoneMemory[i].ContactLastName, PhoneMemory[i].ContactPhoneNumber);
     system("pause");
}
void Search() // Searches for number when given name
{
     int i;
     int q = 0;
     char firstName[25];
     char lastName[25];
     printf("\nEnter Contact's First Name: ");
     scanf("%s", firstName);
     printf("\nEnter Contact's Last Name: ");
     scanf("%s", lastName);
     for (i = 0; i < Count; i++)
     {
         if (strcmp(PhoneMemory[i].ContactFirstName, firstName) == 0 && strcmp(PhoneMemory[i].ContactLastName, lastName) == 0)
         {
            printf("\n%s %s's number is:  %s\n", PhoneMemory[i].ContactFirstName, PhoneMemory[i].ContactLastName, PhoneMemory[i].
ContactPhoneNumber);
            q = 1;
            break;
         }
     }
     if (q != 1)
     {
     printf("\nContact does not exist\n");
     }
     system("pause");
}
void Random() // Randomly selects a contact
{
        srand(time(NULL));
    int Random;
    Random = (rand() % Count) + 1;
    printf("%s %s:  %s\n", PhoneMemory[Random].ContactFirstName, PhoneMemory[Random].ContactLastName, PhoneMemory[Random].ContactPhoneNumber);
    system("pause");

}
```

```
void Reset() // Wipes entire phonebook
{
        int i;
    for (i = 0; i < Count; i++)
    {
        do{ // Deletes all contacts
                PhoneMemory[i].ContactFirstName = NULL;
                PhoneMemory[i].ContactLastName = NULL;
                PhoneMemory[i].ContactPhoneNumber = NULL;
                break;
         }
                 while (i <= Count);
    }
     printf("\nPhonebook has been reset\n");
     system("pause");
}
```