```c
/*
Course: Hardware oriented programming
Assignment: 10
Student: Mads Richardt
Student ID: s224948
*/

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#define MAX_NAME_LENGTH 50
#define MAX_ADDRESS_LENGTH 100
#define MAX_DATA_BASE_LENGTH 100

// Declaration of person struct
typedef struct person
{
        char firstName[MAX_NAME_LENGTH];
        char lastName[MAX_NAME_LENGTH];
        unsigned int age;
        char address[MAX_ADDRESS_LENGTH];
        size_t phoneNumber;
        struct person *nextPtr;
} Person;

// Function declarations
int addPersonToCsvFile(char *filename);
Person *stringToPersonPointer(char *string);
Person *scvFileToLinkedList(char *fileName, size_t *personCount);
int sortLinkedPersonList(Person **startNode, size_t personCount);
Person *swapNodes(Person *node1Ptr, Person *node2Ptr);
Person *getMiddleNode(Person *headNode, size_t length);
Person *linkedListBinarySearch(Person *headNode, size_t value, size_t length);
int viewPerson(Person *person);

int main()
{

        char fileName[MAX_NAME_LENGTH] = "database";
        size_t personCount = 0;
        int controlVar = 0;
        Person *headNode = NULL;
        int sortedFlag = 0;

        puts("************************");
        puts("Welcome to Assignment 9");
        puts("************************");

        // Get file name from user
        printf("\nEnter data base name: ");
        scanf("%s", fileName);

        // Load SCV file into linked list.
        headNode = scvFileToLinkedList(fileName, &personCount);

        // Sort linked list.
        sortLinkedPersonList(&headNode,personCount);
        sortedFlag = 1;


        // Check for format error
        if (personCount > 0 && headNode == NULL)
        {
                printf("Format error on line %lu of csv file %s\n", personCount, fileName);
                printf("Program closing...");
                exit(0);
        }
        // Check if file exits
        if (personCount == 0 && headNode == NULL)
        {
                printf("File could not be opened.");
                printf("Program closing...");
                exit(0);
        }

        do
        {
                printf("\n1: Add person to %s.\n2: Search %s on Phone Number.\n3: Close Program.\nPlease choose option: ", fileName, fileName);
                scanf("%1d", &controlVar);
                puts("");

                switch (controlVar)
                {
                        case 1:
                        {
                                addPersonToCsvFile(fileName);
                                sortedFlag = 0;
                                break;
                        }
                        case 2:
                        {
                                size_t searchNumber;

                                // Get phone number
                                printf("Enter phone number: ");
                                scanf("%lu",&searchNumber);

                                // Sort if not sorted
                                if (sortedFlag == 0)
                                {
                                        personCount = 0;

                                        // Load SCV file into linked list.
                                        headNode = scvFileToLinkedList(fileName, &personCount);

                                        // Sort linked list
                                        sortLinkedPersonList(&headNode, personCount);

                                        // set sorted flag
                                        sortedFlag = 1;
                                }

                                Person *result = linkedListBinarySearch(headNode, searchNumber, personCount);

                                if (result == NULL)
                                {
                                        puts("No match found.");
                                }
                                else {
                                        puts("\nMatch found:");
                                        viewPerson(result);
                                }

                                break;
                        }
                        default:
                                break;
                }

        } while (controlVar != 3);

        // Close program.
        puts("Program closing...");
        return 0;
}

Person *linkedListBinarySearch(Person *headNode, size_t value, size_t length)
{
        Person *startNode = headNode;

        do
        {
                // Get middle node.
                Person *middleNode = getMiddleNode(startNode, length);

                // Return NULL if middle is empty
                if (middleNode == NULL)
                {
                        return NULL;
                }

                // If middleNode contains value, return middleNode
                if (middleNode->phoneNumber == value)
                {
                        return middleNode;
                }

                // If value larger than middleNode->phoneNumber
                if (middleNode->phoneNumber < value)
                {
                        startNode = middleNode->nextPtr;
                        length = length/2;
                }
                else
                {
                        length = length/2;
                }
        } while (1);

        return NULL;
}

int addPersonToCsvFile(char *filename)
{
        // Open file in append mode
        FILE *fPtr = fopen(filename, "a+");

        // Creat Person struct
        Person person;

        // Get first name from user.
        printf("Enter First Name: ");
        scanf("%s", person.firstName);

        // Get last name from user.
        printf("Enter Last Name: ");
        scanf("%s", person.lastName);

        // Get age from user.
        printf("Enter Age: ");
        scanf("%u", &person.age);

        // Get address from user.
        printf("Enter Address: ");
        scanf("%*[\n]%[^\n]", person.address);

        // Get phone number from user.
        printf("Enter Phone Number: ");
        scanf("%lu", &person.phoneNumber);

        // Append person to SCV file.
        fprintf(fPtr, "\n%s, %s, %u, %s, %lu", person.firstName, person.lastName,
person.age, person.address, person.phoneNumber);
        fclose(fPtr);

        return 1;
}

Person *stringToPersonPointer(char *string)
{
        // Declare person pointer.
        Person *person;

        // Initialize format string.
        char *formatString = "%[^,]%*[, ]%[^,]%*[, ]%u%*[, ]%[^,]%*[, ]%zu";

        // Allocate space for Person struct in heap.
        person = (Person *)malloc(sizeof(Person));

        // Scan string into person members.
        int scanned = sscanf(string, formatString, person->firstName, person-
>lastName, &person->age, person->address, &person->phoneNumber);

        // Set nextPtr to NULL.
        person->nextPtr = NULL;

        // Set person to NULL if all members were not scanned correctly.
        if (scanned != 5)
        {
                person = NULL;
        }

        // return Person pointer.
        return person;
}

Person *scvFileToLinkedList(char *fileName, size_t *personCount)
{
        // Declare node pointers.
        Person *startNode, *tempNode, *currentNode;

        // Open file
        FILE *fPtr = fopen(fileName, "r+");

        // If file does not exists return fPtr.
        if (fPtr == NULL)
        {
                return (Person *)fPtr;
        }

        // Declare getline() buffer pointer.
        char *line = NULL;

        // Declare getline() buffer size.
        size_t len = 0;

        // Scan file line by line.
        while ((getline(&line, &len, fPtr)) != -1)
        {
                if (*personCount == 0)
                {
                        // Increment personCount
                        *personCount = *personCount + (size_t)1;

                        // Initialize startNode.

                        startNode = stringToPersonPointer(line);

                        // Return NULL if stringToPersonPointer() did not scan all members correctly.
                        if (startNode == NULL)
                        {
                                // Free line buffer
                                free(line);
                                // Close file
                                fclose(fPtr);
                                return startNode;
                        }

                        tempNode = startNode;
                }
                else
                {
                        // Increment personCount.
                        *personCount = *personCount + (size_t)1;

                        // Initialize currentNode.
                        currentNode = stringToPersonPointer(line);

                        // Return NULL if stringToPersonPointer() did not scan all members correctly.
                        if (currentNode == NULL)
                        {
                                // Free line buffer
                                free(line);
                                // Close file
                                fclose(fPtr);
                                return currentNode;
                        }

                        tempNode->nextPtr = currentNode;
                        tempNode = currentNode;
                }
        }

        // Free line buffer
        free(line);
        // Close file
        fclose(fPtr);

        return startNode;
}

int sortLinkedPersonList(Person **startNode, size_t personCount)
{
        Person **tempNode;
        size_t swapped;
        Person *p1, *p2;

        // Outer loop
        for (size_t i = 0; i <= personCount; i++)
        {

                tempNode = startNode;
                swapped = 0;

                // Inner loop
                for (size_t j = 0; j < personCount - i -1; j++)
                {
                        p1 = *tempNode;
                        p2 = p1->nextPtr;

                        if (p1->phoneNumber > p2->phoneNumber)
                        {
                                *tempNode = swapNodes(p1,p2);
                                swapped = 1;
                        }

                        tempNode = &(*tempNode)->nextPtr;
                }
                if (swapped == 0)
                {
                        break;
                }
        }
        return 1;
}

Person *swapNodes(Person *node1Ptr, Person *node2Ptr)
{
        Person *temp = node2Ptr->nextPtr;
        node2Ptr->nextPtr = node1Ptr;
        node1Ptr->nextPtr = temp;

        return node2Ptr;
}

Person *getMiddleNode(Person *headNode, size_t length)
{
        Person *middleNode = headNode;
        // find middle node
        for (size_t i = 1; i < length / 2; i++)
        {
```

```c
                middleNode = middleNode->nextPtr;
        }
        return middleNode;
}

int viewPerson(Person *person)
{
        printf("Name: %s %s\n", person->firstName,person->lastName);
        printf("Age: %u\n",person->age);
        printf("Address: %s\n",person->address);
        printf("Phone number: %lu\n", person->phoneNumber);
        return 1;
}
```