



# Telephone Book

**Prepared by:**

Name	ID	Group
Youssef Samuel Nachaat Labib	6978	1
Youssef Amr Ismail Othman	6913	2
Arsany Mousa Fathy Rezk	6927	2

# Index

Function Description	Page Number
➤ Load file	4
➤ Phonebook options	6
▪ Input validation	6
▪ Function Selector	6
➤ Back	7
▪ Save	8
▪ Decide	9
➤ Search For a contact	9
➤ Add a contact	10
▪ Enter data	10
○ Phone Number validation	10
○ Date Validation	11
○ Email Validation	11
▪ Entering Data and adding a contact	13
➤ Deleting a contact	14
➤ Modifying a contact	15
➤ Printing the file	16
▪ Swapping	16
▪ Sorting by last name	17
▪ Sorting by date of birth	18
▪ Printing	19
➤ Quit	20

## **User Manual and Sample Runs**

## **Page Number**

- Load file 21
- Search For a contact 22
- Add a contact 23
- Deleting a contact 24
- Modifying a contact 25
- Printing the file 26
- Saving and Quitting 27

# FUNCTIONS DESCRIPTION

## 1) Load File

**loadFile()** function is the first function to be called in our programme.

The user is asked to enter file's name, if file is found it will be opened automatically, and if not, two choices will be available to the user either to enter filename again or create a new file in the same directory with the same name.

For both cases, a validation will take a place to check whether this entry is an integer or not. If not, the user will be prompted to enter his choice again.

```
46 void loadFile()
47 {
48     printf("Please enter the name of the file you want to load: "); //User is prompted to enter the file name and extension
49     scanf("%s", fileName);
50     strcat(fileName, ".txt");
51     FILE*f=fopen(fileName, "r"); //Open the file in a read mode.
52     if(!f)
53     { //If the file can not be opened, the user will be asked whether he would like to re-enter its name or create a new one
54         printf("\nFILE CANNOT BE OPENED! Please choose one of the following options:\n\
55             t(1)\tRe-enter file name\n\t(2)\tCreate a file with this name\nYour choice: ");
56         int z;
57         do{
58             char x[4];
59             z = input(x); //Prompt the user for his choice and check whether it is an integer or not.
60             if (z!=1&& z!=2)
61                 printf("NOT FOUND!\nPlease re-enter your choice: ");
62         }while (z!=1&& z!=2);
63     }
```

Opposingly, 3 cases are to happen:-

First: if he chooses 1, **loadFile()** function will be called again.

Second: if he chooses 2, a message will be displayed informing the user about a successful creation of the file and **saveFile()** function will be called.

Third: if he enters any other number an error message will be displayed to let the user re-enter his choice.

To make sure that we are loading a non-empty file, 2 variables containing the starting and ending positions of the file are compared to each other. If found equal, that means the file is empty as a result, the user will be asked to add a contact and main menu will be displayed.

Else, cursor will be moved using `rewind()` function to the beginning of the file. while loop is used till reaching the end of file to access and read all contacts using the `fscanf()` functions. The second argument in the `fscanf()` function "format" is specific for each data type. The "%[^,]," format is used in the strings: First name, Last name, Address, Phone number to scan all the data till reaching the first comma and skips it. The global variable `n` would be incremented by 1 in each loop to reach the final value (Total number of contacts in the text document). After that, the file is to be closed.

```
63 switch(z)
64 {
65     case 1: loadFile(); break; //Function will be recalled so the user can re-enter his choice.
66     case 2: printf("A new file with the name %s is created in your directory\n", fileName); saveFile(); break;
67             //User will be asked to add a contact.
68     default: break;
69 }
70 }
71 else //If the file exists and can be opened.
72 {
73     int stPos, edPos; //Variables storing start and end position of the cursor in file
74     stPos = ftell(f);
75     fseek(f, 0, SEEK_END);
76     edPos = ftell(f);
77     if (stPos == edPos) { //If the beginning of the file is the same as its end (File is empty), scanning will not be made.
78         rewind(f); //Return cursor to the beginning of the file
79         while (!feof(f)) //Loop until the end of the file, for each contact get all its data and put the contact in the array.
80         {
81             fscanf(f, "%[^,]", c[n].l_name);
82             fscanf(f, "%[^,]", c[n].f_name);
83             fscanf(f, "%d-%d-%d", &c[n].DoB.day, &c[n].DoB.month, &c[n].DoB.year);
84             fscanf(f, "%[^,]", c[n].address);
85             fscanf(f, "%[^,]", c[n].num);
86             fscanf(f, "%s\n", c[n].mail);
87             n++; //n is a global variable starting with 0, is incremented by 1 after each loop,
88                 //so after reaching the end of the file n will be equal to the number of contacts in the file.
89         }
90     else printf("FILE IS EMPTY!\nYou have to add a contact!\n");
91     fclose(f); //Close the file.
92 }
93 }
```

## 2) Phonebook Options

### a) Input validation

`input(char *x)` function is used to validate the integer input of the user to avoid character input. Validation is implemented using `atoi()` function that returns 0 if no validation conversion takes place. Character input in certain parts may result in an infinite loop that will crack the programme.

```
94  int input(char *x)
95  {
96      int i;
97      do{
98          scanf("%s", x); //scan the number entered by the user in a string.
99          i=atoi(x); //Convert the string to a integer.
100         if (i==0) //i will be equal to 0 if no valid conversion takes place.
101             printf("INVALID CHARACTER INPUT!\nPlease re-enter your choice: ");
102         }while(i==0);
103         return i; //The integer is returned
104     }
```

### b) Function selector

`funcselect(int opt)` is the function that will direct the programme according to the user's selection which is saved in the integer variable `opt`.

switch case condition is used accordingly. For instance, if the user enters '1', `searchFile()` function would be called. If the user enters none of the valid options, default case would be applied that will print to the user an error message and ask him to re-enter his choice.

```
105 void funcSelect(int opt)
106 {
107     switch(opt) //Variable "opt" is the choice made by the user, and according to it the function will be called.
108     {
109         case 1: searchFile(); break;
110         case 2: add_contact(); break;
111         case 3: delete_contact(); break;
112         case 4: modify_contact(); break;
113         case 5: printFile(); break;
114         case 6: saveFile(); break;
115         case 7: quit(); break;
116         default: //If the user chooses a number which is not from the options, he will be asked to re-enter the number.
117             printf("INVALID CHOICE!\nPlease re-enter your choice: ");
118             char x[4];
119             opt=input(x);
120             funcSelect(opt);
121             break;
122     }
123 }
```

`phonebookOpts()` function is used to print the menu and calls both functions `input(char *x)` to scan and validate the user's entry and `funcselect(int opt)` to direct the programme according to the user's choice.

```
124 void phonebookOpts()
125 {
126     char x[4];
127     printf("\t\tPhonebook options\n\t(1)\t\tSearch for a contact\n\t(2)\t\tAdd a contact\n\t(3)\t\tDelete a contact\n\t(4)\t\tModify a contact\n\t(5)\t\tPrint phonebook\n\t(6)\t\tSave to file\n\t(7)\t\tExit programme\n\tSelect an option from the menu:\t");
128
129     int opt=input(x); //Prompt the user for his choice, check whether it is an integer.
130     funcSelect(opt); //Call the function selected.
131 }
132
```

### 3) Back

`back()` function asks the user either he wants to go back to menu or not. User would be prompted to enter either 'Y' or 'N'(Not case-sensitive). If he chose 'Y', `phonebookOpts()` function would be called. Else the programme will terminate. Validation was used to assure that the input is either 'Y' or 'N' (Not case-sensitive).

```
133 void back()
134 {
135     printf("\nDo you want to go back to menu(Y or N)? "); //The user is asked if he wants to go back to the main menu.
136     char ch;
137     scanf(" %c",&ch);
138     while((ch!='Y'&&ch!='y')&&(ch!='N'&&ch!='n'))
139     {
140         printf("\nINVALID INPUT!\nPlease re-enter your choice: ");
141         scanf(" %c",&ch);
142     }
143     if(ch=='Y' || ch=='y')
144         phonebookOpts(); //If the user said YES, the menu is displayed.
145     else if(ch=='N' || ch=='n')
146     {
147         printf("\nTHANK YOU!\n"); //If the user said NO, he will exit the programme.
148         exit(0);
149     }
150 }
```

## 4) Deciding Functions

### a) Save File

**saveFile()** function is used to open the file in write only mode, to copy the data in the array to it. If the file was not opened by any means, an error message would be displayed. **fprintf()** function is used to copy the data to the file in a comma separated format. After that, the file is to be closed. Finally, **back()** function is to be called.



```
151 void saveFile()
152 {
153     int i;
154     FILE*f=fopen(fileName,"w"); //Open the file in write mode to re-enter the contacts.
155     if(!f)
156     {
157         printf("FILE CANNOT BE SAVED!");
158         exit(1);
159     } //Exit if the file can not be opened.
160     for(i=0;i<n;i++) // Loop until the end of the array.
161     {
162         fprintf(f,"%s,%s,%d-%d-%d,%s,%s,%s\n",c[i].l_name,c[i].f_name,c[i].DoB.day,c[i].DoB.month,c[i].DoB.year,c[i].address,c[i].num,c[i].mail);
163         //Print all contacts in the file.
164     }
165     fclose(f);
166     if(n)
167         printf("\nDATA SAVED SUCCESSFULLY TO FILE!");
168     back();
169 }
```



## b) Decide

**decide()** function asks the user whether he would like to save the file using **saveFile()** function or go back to menu using the **phonebookOpts()** function, and displays an error message if the user entered an invalid input.

```
170 void decide()
171 {
172     printf("\n\t\tDo you want to save or go back to menu\n\t(1)\t\tSave to file\n\t(2)\t\tGo back to menu\nSelect an option from the menu: ");
173     //Ask the user if he wants to save or quit.
174     int z;
175     do{
176         char x[4];
177         z = input(x); //Prompt the user for his choice and check whether it is an integer or not.
178         if (z!=1&&z!=2)
179             printf("INVALID CHOICE!\nPlease re-enter your choice: ");
180     }while (z!=1&&z!=2);
181     switch(z)
182     {
183         case 1: saveFile(); break; //Changes will be saved in the file.
184         case 2: phonebookOpts(); break; //Main menu will be displayed.
185         default: break;
186     }
187 }
```

## 5) Search for a contact

**searchFile()** function asks the user to enter the last name for the contact he wants to search for.

**ALGORITHM (Linear search):** User's entry would be compared to each last name stored in the array using a for loop. If a match is found, the contact's record would be displayed for all matches. Otherwise, a message would be displayed to the user informing him/her that no matches were found.

```
205 void searchFile()
206 { //Linear Search
207     char cname[20];
208     printf("\nSEARCH FOR A CONTACT\nPlease enter contact last name:\t");
209     scanf("%s", cname); //The last name the user wants to search is stored in variable cname.
210     int i, cnt=0;
211     for(i=0; i<n; i++) //Loop on all the contacts of file.
212     {
213         if(strcmp(c[i].l_name, cname)==0) //When the contact the user searched is found, the data of this contact will be printed.
214         {
215             printf("\nContact %d\nFirst name: %s\tAddress: %s\tEmail: %s\tPhone number: %s\n", cnt+1, c[i].f_name, c[i].address, c[i].mail, c[i].num);
216             cnt++; //When a contact with this last name is found, this variable is incremented by 1.
217         }
218     }
219     if(cnt==0) //If this variable equals to 0, that means that it has not been incremented, so no contacts are found.
220         printf("\nNO MATCHES FOUND!\n");
221     back(); //The user is asked if he wants to go back to menu.
222 }
```

## 6) Adding a contact

### a) Enter data

- **Phone number validation:** `checkNum(char nmb[16], int except)` function is used to validate the phone numbers entered by the user. 3 types of validations are used:
  - First: we check that the phone number length is between 6 and 15 (Standard Length).
  - Second: we check that the phone number entered is not of another existing contact. Variable (except) is the index for the contact that is being added or modified, we must not compare it with itself.
  - Third: the programme checks for the user entry to assure that the input is only digits (No special characters or letters are allowed). That check is made using the ASCII digits boundaries.

For any wrong entry, and error message would be displayed to the user and he would be prompted for re-entry.

```
223 int checkNum(char nmb[16], int except) //the input of this function is the phone number to check,
224 //and except which is the position of the contact to check.
225 {
226     int i,nl=strlen(nmb);
227     if(nl<6||nl>15)
228     {
229         printf("INVALID PHONE NUMBER LENGTH!\n");
230         return 1;
231     }
232     for(i=0; i<nl; i++) //Check that the phone number consists only of digits (0-9).
233     {
234         if(nmb[i]<48||nmb[i]>57) //According to the ASCII table, a digit is between 48 and 57,
235             //so any character in the phone number out of this range is incorrect.
236         {
237             printf("INVALID CHARACTER INPUT!\n");
238             return 1;
239         }
240     }
241     for(i=0; i<n&&i!=except; i++) //Check whether another contact has the same phone number.
242     {
243         if(strcmp(c[i].num, nmb)==0) {
244             printf("NUMBER ALREADY EXISTS!\n");
245             return 1;
246         }
247     }
248     return 0; //Returns 0 if the phone number is valid.
```

- **Date validation:** `checkDate(int d, int m, int y)` function validates the date entered by the user by checking the year to be smaller than 2021, month to be from 1 to 12 inclusive and finally day to be from 1 to 31 except for months (4-6-9-11) in which they have 30 days only.

It checks also if months was February so it has a maximum of 29 days unless it was a leap year where maximum number of days will be 28.

```

249 int checkDate(int d,int m,int y) //d, m and y are the day, month and year entered by the user respectively.
250 {
251     if (y>2021) //Check that the year has passed.
252     {
253         printf("INVALID YEAR\n");
254         return 1;
255     }
256     if (m<1 || m>12) //Check that the month is between 1 and 12.
257     {
258         printf("INVALID MONTH\n");
259         return 1;
260     }
261     if (d<1 || d>31) //Check that the day is between 1 and 31.
262     {
263         printf("INVALID DAY\n");
264         return 1;
265     }
266     if (d==31 && (m==4 || m==6 || m==9 || m==11)) //Check that there is 31 days in this month.
267     {
268         printf("THIS MONTH DOES NOT HAVE 31 DAYS\n");
269         return 1;
270     }
271     if (m==2 && d>29) //Check that February has not more than 29 days.
272     {
273         printf("FEBRUARY DOES NOT HAVE MORE THAN 29 DAYS\n");
274         return 1;
275     }
276     if (m==2 && d==29 && (y%400 != 0 && (y%4!=0 || y%100==0))) //Check if it is a leap year and February has 29 days.
277     {
278         printf("FEBRUARY ONLY HAS 28 DAYS IN THIS YEAR\n");
279         return 1;
280     }
281     return 0; //It is valid date when all the previous conditions were false.
282 }

```

- **Email validation:** `check_mail(char x[], int except)` function validates the email entered by the user. 3 types of validations are used:
  - First: we check that the email entered is not of another existing contact. Variable (except) is the index for the contact that is being added or modified, we must not compare it with itself.

- Second: we check that the username and domain do contain only ‘\_’, ‘-’, ‘.’, ‘A to Z’, ‘a to z’, ‘0 to 9’.
- Third: we check that the domain ends with .com and it consists of more than 4 characters.

For any wrong entry, and error message would be displayed to the user and he would be prompted for re-entry.

```

283 int check_mail(char x[], int except) //x is the mail entered by the user, except is the position of the contact to check.
284 {
285     int i,length=strlen(x),j,found=0;
286     char domain[15]; //Checking that the domain is valid.
287     for (i=0;i<n;i++) //Check whether another contact has the same email.
288     {
289         if(!strcmp(x,c[i].mail)&&i!=except)
290         {
291             printf("EMAIL ALREADY EXISTS!\n");
292             return 1; //Invalid email.
293         }
294     }
295     for(i=0;i<length;i++) //Find the position of '@' in the email address.
296     {
297         if(x[i]=='@')
298         {
299             j=i; //j is the index of '@'
300             found=1;
301             break;
302         }
303     }
304     if(!found)
305     {
306         printf("INVALID INPUT\n");
307         return 1;
308     }
309     for(i=0;i<length;i++) //Checking that the email address before only consists of digits, characters (a-z) or (A-Z), underscore or '.' or '-'.
310     {
311         if((i!=j) && !(x[i]=='_'||x[i]=='.'||x[i]=='-'||(x[i]>47&&x[i]<58)|| (x[i]>64&&x[i]<91)|| (x[i]>96&&x[i]<123)))
312         {
313             printf("INVALID INPUT!\n");
314             return 1;
315         }
316     }
317     strcpy(domain, x+j+1); //Copy the part of the email address after the '@'.
318     if(strcmp(x+length-4, ".com") != 0 || (strlen(domain)<6)) //Checking that the email ends with ".com" and that the domain length is bigger than 4.
319     {
320         printf("INVALID DOMAIN!\n");
321         return 1;
322     }
323     return 0; //Email is valid.
324 }

```

`enter_data(int x)` function prompts the user to enter all required fields for the contact. It copies the entered data to the array at index 'x' which is the first empty field in the array. The function uses all validation functions mentioned above.

```

325 void enter_data(int x) //x is the position the contact added or modified
326 {
327     //User is prompted field by field for the data of the new contact.
328     printf("First name: ");
329     scanf("%s", c[x].f_name);
330     printf("Last name: ");
331     scanf("%s", c[x].l_name);
332     printf("Birth date(day-month-year): ");
333     scanf("%d-%d-%d", &c[x].DoB.day, &c[x].DoB.month, &c[x].DoB.year);
334     int w=checkDate(c[x].DoB.day,c[x].DoB.month,c[x].DoB.year); //Validation of the date.
335     while(w) //While is true when "w" != 0, that means that the date entered was incorrect.
336     {
337         printf("Please re-enter birth date(day-month-year): ");
338         scanf("%d-%d-%d", &c[x].DoB.day, &c[x].DoB.month, &c[x].DoB.year);
339         w=checkDate(c[x].DoB.day,c[x].DoB.month,c[x].DoB.year);
340     }
341     printf("Address: ");
342     scanf(" %s", c[x].address);
343     printf("Phone number: ");
344     scanf("%s", c[x].num);
345     w=checkNum(c[x].num,x);
346     while(w) //While is true when "w" != 0, that means that the phone number entered was incorrect.
347     {
348         printf("Please re-enter phone number: ");
349         scanf("%s",c[x].num);
350         w=checkNum(c[x].num,x);
351     }
352     printf("Email: ");
353     scanf("%s", c[x].mail);
354     w=check_mail(c[x].mail, x);
355     while(w) //While is true when "w" != 0, that means that the email entered was incorrect.
356     {
357         printf("Please re-enter the email: ");
358         scanf("%s", c[x].mail);
359         w=check_mail(c[x].mail, x);
360     }
361 }

```

`addcontact()` function calls the `enter_data(int x)` function and increments the total number of contacts by 1. `decide()` function is to be called.

```

362 void add_contact()
363 {
364     printf("\nADD CONTACT\nPlease enter data of the new contact: \n");
365     enter_data(n); //Here the user will be asked to enter all data of the new contact.
366     n++; //Global variable n is incremented by 1, because a new contact was added.
367     decide(); //Ask the user if he wants to save or go back to menu.
368 }

```

## 7) Deleting a contact

`delete_contact()` function prompts the user to enter first and last names, all records associated with the names should be deleted. User's entry would be compared to each contact stored in the array using a for loop. If a match is found, the contact's index number would be stored in the array "index[n]" and his/her data would be displayed on the screen. Variable 'k' will have the number of matches. If no matches were found, a message will appear to the user stating so and `back()` function will be called .

A for loop is used to shift all the contacts after the one to be deleted replacing each contact by the one after it resulting in contact deletion. Consequently, all indices in the "index[n]" array should be decremented by 1.

The shifting will happen according to the number of matches (Variable 'k') and (Variable 'n') is to be decremented by k. `decide()` function is to be called thereafter.

```
369 void delete_contact()
370 {
371     int i,j,k=0,index[n],m=0,z=0;
372     char cfname[20],clname[20];
373     printf("\nDELETE CONTACT\nPlease enter contact first name: "); //Prompt the user for the first and last name of the contact he wants to delete.
374     scanf("%s",cfname);
375     printf("Please enter contact last name: ");
376     scanf("%s",clname);
377     for (i=0;i<n;i++) //Looping on all contacts, searching for contacts with these first and last name.
378     {
379         if (strcmp(c[i].l_name,clname)==0&&strcmp(c[i].f_name,cfname)==0) //When a contact is found, it will be displayed to the user.
380         {
381             printf("\nContact to be deleted:\nDate of Birth: %d-%d-%d\tPhone number: %s\n",c[i].DoB.day,c[i].DoB.month,c[i].DoB.year,c[i].num);
382             index[k] = i; //index is an array of integers that will contain the index of each contact to be deleted.
383             k++; //k starting by 0, incremented by 1 each time a contact is found, so will finally represent the number of contacts to be deleted.
384         }
385     }
386     if (!k) //if k=0, that means that no contacts were found.
387     {
388         printf("\nNO CONTACTS FOUND!\n");
389         back(); //Ask the user if he wants to go back to menu.
390     }
391     else
392     {
393         for(i=0;i<k;i++) //Looping a number of time equals the number of contacts to be deleted.
394         {
395             for(j=index[m]-z;j<n-1;j++) //Start from the first contact to be deleted, replace each contact by the one after it.
396             {
397                 c[j]=c[j+1];
398             }
399             z++; //z is incremented by 1 because all the contacts were shifted to the left,
400             //so the next time in the loop we want to start from the index of the next contact to be deleted but decremented by 1.
401             n--;m++; //n is decremented by 1 because a contact was deleted, m is incremented to get the index of the next contact to be deleted.
402         }
403         decide(); //Decide to save or go back to menu.
404     }
405 }
```



## 8) Modifying a contact

**modify\_contact()** function prompts the user to enter the last name, all records associated with the name will be displayed. If no contacts are found a message will be displayed informing the user that no contacts were found and **back()** function is to be called .

The user is asked to choose the contact he wants to modify and his choice is going to be validated using **input(char \*x)** function, after that function **enter\_data()** is called to allow the user entering his new details regarding that contact. **decide()** function is to be called thereafter.

```
406 void modify_contact()
407 {
408     int i, index[n], k=0, x, j;
409     char cname[20];
410     printf("\nMODIFY CONTACT\nPlease enter contact last name:\t"); //Prompt the user for the last name of the contact he wants to modify.
411     scanf("%s", cname);
412     for(i=0; i<n; i++) //Looping on all contacts, searching for contacts with this last name.
413     {
414         if (strcmp(c[i].l_name, cname)==0)
415         {
416             index[k]=i; //index is an array of integers that will contain the index of each contact to be modified.
417             k++; //k starting by 0, incremented by 1 each time a contact is found, so will finally represent the number of contacts to be modified.
418         }
419     }
420     if(!k) //if k=0, that means that no contacts were found.
421     {
422         printf("\nNO CONTACTS FOUND!\n");
423         back(); //Ask the user if he wants to go back to menu.
424     }
425     else
426     {
427         if(k>1) //If there is more than one contact with this last name, display these contacts and let the user choose which one does he want to modify.
428         {
429             for(i=0; i<k; i++)
430             {
431                 printf("\n%d) %s %s\tEmail: %s\tPhone Number: %s\n", i+1, c[index[i]].f_name, c[index[i]].l_name, c[index[i]].mail, c[index[i]].num);
432                 //Data of contacts with this last name.
433             }
434             printf("\nPlease choose which contact do you want to modify: ");
435             char p[4];
436             do{
437                 j=input(p); //The user is prompted for his choice.
438                 if (j>k) //If the user entered a number higher than the possible ones.
439                     printf("INVALID CHOICE!\nPlease re-enter your choice: ");
440             }while(j>k); //Loop until he choose a valid number.
441             x=index[j-1]; //x represents the index of the contact to be modified.
442         }
443         else
444         {
445             x=index[0]; //If only one contact was found with this last name, x will be equal to the first and only element in the array.
446         }
447         printf("\nContact to be modified: \n%s %s\tDate of Birth: %d-%d-%d\tEmail: %s\tAddress: %s\n",
448             c[x].f_name, c[x].l_name, c[x].DoB.day, c[x].DoB.month, c[x].DoB.year, c[x].mail, c[x].address, c[x].num);
449         printf("\nModify the data: \n");
450         enter_data(x); //Prompting the user field by field for the data of the contact he wants to modify.
451         decide(); //Ask the user if he wants to save or go back to menu.
452     }
453 }
```

## 9) Printing the directory

### a) Swapping

`swap (Contact *x, Contact *y)` function has the argument for the addresses of 2 contacts to swap them. The function initializes another variable of struct type "Contact" to aid in swapping.

The function has a return value of '0'.

```
454 int swap (Contact *x, Contact *y)
455 {
456     //Swap function takes as input the addresses of 2 contacts and returns 0.
457     Contact temp;
458     temp = *x;
459     *x = *y;
460     *y = temp;
461     return 0;
462 }
```





## b) Sorting by last name

`sortByLname()` function sorts the directory in an ascending order of last name.

**ALGORITHM (Bubble sort):** starts by comparing the first contact "last name" with the second contact "last name", if the first element is greater than the second element, it will swap both elements using the `swap()` function, and then move on to compare the second and the third element, and so on. In case that 2 contacts have the same last name, they will be sorted according to their first names.

Comparison for the 2 strings is made using `strcmp()` function. sorted variable acts as a flag to reduce execution time if the array was sorted. If sorted remained 1 in any loop, the loop would be terminated, else the variable sorted will have the return value of `swap()` function '0'.

```
463 void sortByLname()
464 { //Bubble Sort
465     int pass,i,sorted=0;
466     for (pass=1;!sorted&&pass<n;pass++) //Loop until you reach the end of the array or when sorted = 1.
467     {
468         sorted=1; //When the array is sorted, 'sorted' will remain = 1, and the loop will end.
469         for (i=0;i<n-pass;i++)
470         {
471             if (strcmp(c[i].l_name, c[i+1].l_name)==1) //When contact 2 is less than contact 1, they will be swapped.
472                 sorted=swap(&c[i],&c[i+1]); //Swap function returns 0 when executed.
473             if (strcmp(c[i].l_name, c[i+1].l_name)==0) //When two contacts have the same last name, they will be sorted by first name.
474                 if (strcmp(c[i].f_name, c[i+1].f_name)==1)
475                     sorted=swap(&c[i],&c[i+1]);
476         } //After this loop, the higher contact in order will be the last one.
477     }
478 }
```

### c) Sorting by Date of Birth

**sortByDoB()** function sorts the directory in an ascending order of the date of birth. It uses bubble sorting as well to execute so. The function compares the year of birth of the first 2 contacts, swapping will take place if contact 1 is higher than contact 2. If the birth year is the same, it checks and compares the month of birth, contacts will be swapped if the month of the first contact is higher than that of the second. If the birth month is the same, it checks and compares the day of birth to swap if the day of birth of contact 1 is higher than that of contact 2. Loops are used to access all contacts if found not sorted, while if sorted variable remained to be 1 the loops will terminate.

```
479 void sortByDoB()
480 { //Bubble Sort
481     int i,j,sorted=0;
482     for(i=1;!sorted&&i<n;i++) //Loop until you reach the end of the array or when sorted = 1.
483     {
484         sorted = 1; //When the array is sorted, 'sorted' will remain = 1, and the loop will end.
485         for(j=0;j<n-i;j++)
486         {
487             if(c[j].DoB.year>c[j+1].DoB.year) //Check year, if contact 1 higher than contact 2, they will be swapped.
488                 sorted=swap(&c[j],&c[j+1]);
489             else if(c[j].DoB.year==c[j+1].DoB.year)
490             {
491                 if(c[j].DoB.month>c[j+1].DoB.month) //If they have the same year, but contact 1 month higher than contact 2, they will be swapped.
492                     sorted=swap(&c[j],&c[j+1]);
493                 else if(c[j].DoB.month==c[j+1].DoB.month)
494                     if(c[j].DoB.day>c[j+1].DoB.day) //If they have the same year and month, but contact 1 day higher than contact 2, they will be swapped.
495                         sorted=swap(&c[j],&c[j+1]);
496             }
497         } //After this loop, the higher contact in order will be the last one.
498     }
499 }
```

## d) Printing

`printFile()` function firstly asks the user to enter his choice of sorting either by last name or date of birth. User will input either '1' to sort by last name (calling `sortByLname()` function) or '2' to sort by date of birth (calling `sortByDoB()` function) .

The input will be validated using the `input(char *x)` to assure that it is an integer. The value would be compared to '1' and '2' if it was not one of those cases, the user would be asked to re-enter his sorting choice. A loop would then be made to pass by all the contacts in the array in which they are sorted according to the user preference to print them. Finally, `decide()` function would then be called.

```

500 void printFile()
501 {
502     if (n==0) //If the file is empty.
503     {
504         printf("\nNO CONTACTS FOUND.\n");
505         back();
506     }
507     else
508     {
509         //Ask the user if he wants to print the contacts sorted by last name or date of birth.
510         printf("\nPRINTING FILE\n\t\tPrinting options\n\t(1)\t\tSorted by last name\n\t(2)\t\tSorted by date of birth\n\tSelect an option from the menu: ");
511         int z;
512         do{
513             char x[4];
514             z = input(x); //Prompt the user for his choice.
515             if (z!=1&&z!=2)
516                 printf("INVALID CHOICE!\nPlease re-enter printing option: ");
517         }while (z!=1&&z!=2); //Loop if the user entered invalid number.
518         switch(z)
519         {
520             case 1: sortByLname(); break; //Sort them by last name.
521             case 2: sortByDoB(); break; //Sort them by date of birth.
522             default: break;
523         }
524         int i;
525         for(i=0; i<n; i++) //After sorting contacts, print them.
526         {
527             printf("Contact %d\nFirst name: %s Last name: %s Birth date: %d-%d-%d Address: %s Phone: %s Email: %s\n",
528                 i+1, c[i].f_name, c[i].l_name, c[i].DoB.day, c[i].DoB.month, c[i].DoB.year, c[i].address, c[i].num, c[i].mail);
529         }
530         decide(); //Ask the user if he wants to save or go back to menu.
531     }
532 }

```

## 10) Quit

**quit()** function is used to exit the programme without saving the data. A warning message is displayed to the user to let him/her confirm that the changes would be discarded.

The user is prompted to enter his choice whether he would like to quit by entering 'Y' or not by entering 'N' (Not case-sensitive). If he chose 'N', **back()** function is to be called. Validation was used to assure that the input is either 'Y' or 'N' (Not case-sensitive).

```
188 void quit()
189 {
190     char ch;
191     printf("\nAll changes will be discarded unless you saved it before\nAre you sure you want to quit(Y or N)? ");
192     scanf(" %c",&ch);
193     while((ch!='Y'&&ch!='y')&&(ch!='N'&&ch!='n')) //If the user wants to quit he is only allowed to type 'Y' or 'y', if not he can type 'N' or 'n'.
194     {
195         printf("INVALID INPUT!\nPlease re-enter your choice: ");
196         scanf(" %c",&ch);
197     }
198     if(ch=='Y' || ch=='y'){ //If the user said YES, changes will not be saved.
199         printf("\nTHANK YOU!\n");
200         exit(0);
201     }
202     else //If the user said NO, he will be asked whether he would like to go back to menu or not.
203         back();
204 }
```

# USER MANUAL AND SAMPLE RUNS

## *How to load a file*

When you open the programme you are prompted to enter the name of the text file you want to load. If there exists a file with the same name in your

directory, the main menu will be displayed where you can (Search, add, delete, modify a contact print your phonebook, save your file, or quit the programme).

Else if the file does not exist, you will be asked either to re-enter the file name or to create a new file with the same name. This choice is made by typing either '1' or '2' respectively.

In case you create a new file, the file will be created in the same directory and will be empty, so it is recommended to add a contact.

```
"C:\Users\ALEX STORE\Desktop\Programming Project Final\main.exe"
Please enter the name of the file you want to load: phonebook
Phonebook options
(1) Search for a contact
(2) Add a contact
(3) Delete a contact
(4) Modify a contact
(5) Print phonebook
(6) Save to file
(7) Exit programme
Select an option from the menu:
```

```
"C:\Users\ALEX STORE\Desktop\Programming Project Final\main.exe"
Please enter the name of the file you want to load: alpha
FILE CANNOT BE OPENED! Please choose one of the following options:
(1) Re-enter file name
(2) Create a file with this name
Your choice: 1
Please enter the name of the file you want to load: Beta
FILE CANNOT BE OPENED! Please choose one of the following options:
(1) Re-enter file name
(2) Create a file with this name
Your choice: 2
A new file with the name Beta.txt is created in your directory
Do you want to go back to menu(Y or N)? Y
Phonebook options
(1) Search for a contact
(2) Add a contact
(3) Delete a contact
(4) Modify a contact
(5) Print phonebook
(6) Save to file
(7) Exit programme
Select an option from the menu:
```

## *How to search for a contact*

Choose searching option by typing 1, then you have to write the last name of the contact you want to search for. The first name, address, email, phone number will be displayed for all matches.

After searching, you will be asked either to go back to menu by entering 'Y' or to exit by entering 'N'.

```
"C:\Users\ALEX STORE\Desktop\Programming Project Final\main.exe"
Please enter the name of the file you want to load: phonebook
Phonebook options
(1) Search for a contact
(2) Add a contact
(3) Delete a contact
(4) Modify a contact
(5) Print phonebook
(6) Save to file
(7) Exit programme
Select an option from the menu: 1

SEARCH FOR A CONTACT
Please enter contact last name: David

NO MATCHES FOUND!

Do you want to go back to menu(Y or N)? y
Phonebook options
(1) Search for a contact
(2) Add a contact
(3) Delete a contact
(4) Modify a contact
(5) Print phonebook
(6) Save to file
(7) Exit programme
Select an option from the menu: 1

SEARCH FOR A CONTACT
Please enter contact last name: Vindahl

Contact 1
First name: Thomas      Address: 362 11th street brooklyn      Email: tvindahl@gmail.com      Phone number: 01208877955

Do you want to go back to menu(Y or N)? Y
Phonebook options
(1) Search for a contact
(2) Add a contact
(3) Delete a contact
(4) Modify a contact
(5) Print phonebook
(6) Save to file
(7) Exit programme
Select an option from the menu: 1

SEARCH FOR A CONTACT
Please enter contact last name: Simpson

Contact 1
First name: Bethany      Address: 11 greenway plaza houston tx      Email: bsimpson@gmail.com      Phone number: 01255987895

Contact 2
First name: Philippe      Address: 33 hudson street jersey city      Email: psimpson@gmail.com      Phone number: 01248488895

Do you want to go back to menu(Y or N)? N


THANK YOU!

Process returned 0 (0x0)   execution time : 51.578 s
Press any key to continue.
```

## *How to add a contact*

Choose adding option by typing 2, then you have to write first name, last name, date of birth in the format DD-MM-YYYY, address, phone number, email.

After that, you have to choose either to save the file by entering '1' or to go back to menu by entering '2'.

 "C:\Users\ALEX STORE\Desktop\Programming Project Final\main.exe"

```
Please enter the name of the file you want to load: phonebook
Phonebook options
(1) Search for a contact
(2) Add a contact
(3) Delete a contact
(4) Modify a contact
(5) Print phonebook
(6) Save to file
(7) Exit programme
Select an option from the menu: 2

ADD CONTACT
Please enter data of the new contact:
First name: John
Last name: Henderson
Birth date(day-month-year): 2-2-2022
INVALID YEAR
Please re-enter birth date(day-month-year): 2-13-2020
INVALID MONTH
Please re-enter birth date(day-month-year): 32-12-2020
INVALID DAY
Please re-enter birth date(day-month-year): 31-4-2020
THIS MONTH DOES NOT HAVE 31 DAYS
Please re-enter birth date(day-month-year): 30-2-2020
FEBRUARY DOES NOT HAVE MORE THAN 29 DAYS
Please re-enter birth date(day-month-year): 29-2-2019
FEBRUARY ONLY HAS 28 DAYS IN THIS YEAR
Please re-enter birth date(day-month-year): 29-2-2020
Address: 318 W Alameda Avenue
Phone number: 30345566
NUMBER ALREADY EXISTS!
Please re-enter phone number: 333
INVALID PHONE NUMBER LENGTH!
Please re-enter phone number: 313abc567
INVALID CHARACTER INPUT!
Please re-enter phone number: 313567891
Email: mcanfield@icloud.com
EMAIL ALREADY EXISTS!
Please re-enter the email: mcan$fiend@gmail.com
INVALID INPUT!
Please re-enter the email: mcanfield@hotmail.net
INVALID DOMAIN!
Please re-enter the email: mcanfieldoutlook.com
INVALID INPUT
Please re-enter the email: mcanfield@a.com
INVALID DOMAIN!
Please re-enter the email: john.Hendy@yahoo.com

Do you want to save or go back to menu
(1) Save to file
(2) Go back to menu
Select an option from the menu: 1

DATA SAVED SUCCESSFULLY TO FILE!
Do you want to go back to menu(Y or N)? n


THANK YOU!

Process returned 0 (0x0) execution time : 320.718 s
Press any key to continue.
```

## *How to delete a contact*

Choose the deleting option by typing '3' and then type first name and last name for the contact you want to delete.

After that, you have to choose either to save the file by entering '1' or to go back to menu by entering '2'.

 "C:\Users\ALEX STORE\Desktop\Programming Project Final\main.exe"

```
Phonebook options
(1)      Search for a contact
(2)      Add a contact
(3)      Delete a contact
(4)      Modify a contact
(5)      Print phonebook
(6)      Save to file
(7)      Exit programme
Select an option from the menu: 3

DELETE CONTACT
Please enter contact first name: Thiago
Please enter contact last name: Alcantara

NO CONTACTS FOUND!

Do you want to go back to menu(Y or N)? Y
Phonebook options
(1)      Search for a contact
(2)      Add a contact
(3)      Delete a contact
(4)      Modify a contact
(5)      Print phonebook
(6)      Save to file
(7)      Exit programme
Select an option from the menu: 3

DELETE CONTACT
Please enter contact first name: Steve
Please enter contact last name: Graves

Contact to be deleted:
Date of Birth: 1-12-1982      Phone number: 3046646454

Do you want to save or go back to menu
(1)      Save to file
(2)      Go back to menu
Select an option from the menu: 3
INVALID CHOICE!
Please re-enter your choice: a
INVALID CHARACTER INPUT!
Please re-enter your choice: 2
Phonebook options
(1)      Search for a contact
(2)      Add a contact
(3)      Delete a contact
(4)      Modify a contact
(5)      Print phonebook
(6)      Save to file
(7)      Exit programme
```



## *How to modify a contact*

**Choose modifying option by typing '4' and then, type the last name of the contact you want to modify. If there is more than one match, these contacts will be displayed, and you will be asked to enter your preference by typing its number. After that, you will be prompted field by field to modify the information. Finally, choose whether you want to save or back to menu.**

```
"C:\Users\ALEX STORE\Desktop\Programming Project Final\main.exe"
Please enter the name of the file you want to load: phonebook
Phonebook options
(1) Search for a contact
(2) Add a contact
(3) Delete a contact
(4) Modify a contact
(5) Print phonebook
(6) Save to file
(7) Exit programme
Select an option from the menu: 4

MODIFY CONTACT
Please enter contact last name: Jason

(1) Michael Jason      Email: jmichael@icloud.com      Phone Number: 0100050506
(2) Mark Jason  Email: mjason@gmail.com Phone Number: 01232165564

Please choose which contact do you want to modify: 3
INVALID CHOICE!
Please re-enter your choice: 1

Contact to be modified:
Michael Jason  Date of Birth: 10-3-1984      Email: jmichael@icloud.com      Address: 21 Drydock Ave #410      Phone Number: 0100050506

Modify the data:
First name: Michael
Last name: Jackson
Birth date(day-month-year): 10-2-1984
Address: 21 Drydock Ave #410
Phone number: 0100050506
Email: jmichael@icloud.com

Do you want to save or go back to menu
(1) Save to file
(2) Go back to menu
Select an option from the menu: 1

DATA SAVED SUCCESSFULLY TO FILE!
Do you want to go back to menu(Y or N)? N

THANK YOU!

Process returned 0 (0x0)   execution time : 183.432 s
Press any key to continue.
```

## *How to print the file*

**Choose printing option by typing '5', then you have to choose either to print it sorted by last name or by date of birth by typing '1' or '2' respectively.**

```
"C:\Users\ALEX STORE\Desktop\Programming Project Final\main.exe"
Please enter the name of the file you want to load: phonebook

Phonebook options
(1) Search for a contact
(2) Add a contact
(3) Delete a contact
(4) Modify a contact
(5) Print phonebook
(6) Save to file
(7) Exit programme
Select an option from the menu: 5

PRINTING FILE
Printing options
(1) Sorted by last name
(2) Sorted by date of birth
Select an option from the menu: 1

Contact 1
First name: Mark Last name: Adam Birth date: 12-10-1990 Address: 48 washington ave chelsea ma Phone: 01231213588 Email: mark.adam@icloud.com
Contact 2
First name: Mark Last name: Brian Birth date: 10-12-1990 Address: 160 W Addison St. Phone: 01565465468 Email: mcanfield@icloud.com
Contact 3
First name: Bethany Last name: Cook Birth date: 3-10-1983 Address: 13 texas street jersey city Phone: 30345566 Email: bcook@hotmail.com
Contact 4
First name: Daniel Last name: Graves Birth date: 2-12-1982 Address: 48 washington ave chelsea ma Phone: 303554564 Email: dgraves@outlook.com
Contact 5
First name: Steve Last name: Graves Birth date: 1-12-1982 Address: 47 newbury st peabody ma Phone: 3046646454 Email: sgraves@yahoo.com
Contact 6
First name: Mark Last name: Jason Birth date: 10-4-1988 Address: 23 sip Ave #410w Phone: 01232165564 Email: mjason@gmail.com
Contact 7
First name: Michael Last name: Jason Birth date: 10-3-1984 Address: 21 Drydock Ave #410 Phone: 0100050506 Email: jmichael@icloud.com
Contact 8
First name: Bethany Last name: Simpson Birth date: 1-11-1982 Address: 11 greenway plaza houston tx Phone: 01255987895 Email: bsimpson@gmail.com
Contact 9
First name: Philippe Last name: Simpson Birth date: 10-10-1981 Address: 33 hudson street jersey city Phone: 01248488895 Email: psimpson@gmail.com
Contact 10
First name: Thomas Last name: Vindahl Birth date: 1-10-1982 Address: 362 11th street brooklyn Phone: 01208877955 Email: tvindahl@gmail.com

Do you want to save or go back to menu
(1) Save to file
(2) Go back to menu
Select an option from the menu: 1

DATA SAVED SUCCESSFULLY TO FILE!
Do you want to go back to menu(Y or N)? N

THANK YOU!

Process returned 0 (0x0) execution time : 15.446 s
Press any key to continue.
```

**After that, you can either save the file in a sorted form according to your choice in printing or go back to menu.**


```
PRINTING FILE
Printing options
(1) Sorted by last name
(2) Sorted by date of birth
Select an option from the menu: 2

Contact 1
First name: Philippe Last name: Simpson Birth date: 10-10-1981 Address: 33 hudson street jersey city Phone: 01248488895 Email: psimpson@gmail.com
Contact 2
First name: Thomas Last name: Vindahl Birth date: 1-10-1982 Address: 362 11th street brooklyn Phone: 01208877955 Email: tvindahl@gmail.com
Contact 3
First name: Bethany Last name: Simpson Birth date: 1-11-1982 Address: 11 greenway plaza houston tx Phone: 01255987895 Email: bsimpson@gmail.com
Contact 4
First name: Steve Last name: Graves Birth date: 1-12-1982 Address: 47 newbury st peabody ma Phone: 3046646454 Email: sgraves@yahoo.com
Contact 5
First name: Daniel Last name: Graves Birth date: 2-12-1982 Address: 48 washington ave chelsea ma Phone: 303554564 Email: dgraves@outlook.com
Contact 6
First name: Bethany Last name: Cook Birth date: 3-10-1983 Address: 13 texas street jersey city Phone: 30345566 Email: bcook@hotmail.com
Contact 7
First name: Michael Last name: Jason Birth date: 10-3-1984 Address: 21 Drydock Ave #410 Phone: 0100050506 Email: jmichael@icloud.com
Contact 8
First name: Mark Last name: Jason Birth date: 10-4-1988 Address: 23 sip Ave #410w Phone: 01232165564 Email: mjason@gmail.com
Contact 9
First name: Mark Last name: Adam Birth date: 12-10-1990 Address: 48 washington ave chelsea ma Phone: 01231213588 Email: mark.adam@icloud.com
Contact 10
First name: Mark Last name: Brian Birth date: 10-12-1990 Address: 160 W Addison St. Phone: 01565465468 Email: mcanfield@icloud.com
```

## *Saving and quitting from the main menu*

After doing any of the previous processes you always had the ability to go back to main menu and choose either save or quit by typing '6' or '7' respectively.

**TAKE CARE:** If you did not save your work and you chose to quit from the main menu, all the changes you made will be discarded.

 "C:\Users\ALEX STORE\Desktop\Programming Project Final\main.exe"

```
Phonebook options
(1)      Search for a contact
(2)      Add a contact
(3)      Delete a contact
(4)      Modify a contact
(5)      Print phonebook
(6)      Save to file
(7)      Exit programme
Select an option from the menu: 6

DATA SAVED SUCCESSFULLY TO FILE!
Do you want to go back to menu(Y or N)? y
Phonebook options
(1)      Search for a contact
(2)      Add a contact
(3)      Delete a contact
(4)      Modify a contact
(5)      Print phonebook
(6)      Save to file
(7)      Exit programme
Select an option from the menu: 7

All changes will be discarded unless you saved it before
Are you sure you want to quit(Y or N)? Y

THANK YOU!

Process returned 0 (0x0)   execution time : 27.174 s
Press any key to continue.
```