Bank Management System Report

December 31, 2023

Done By:

Ahmed Wael Mohamed Gaber	*ID: 8836*
Ahmad Zaki Ahmad Zaki Mahrous	*ID: 8910*
Asser Mohamed Ahmed Hanafy Mahmoud	*ID: 8833*
Seifeldin Ahmed Maazouz Zeid Ismail	*ID: 8927*
Abdelrahman Ahmed Mohamed Agha	*ID: 8918*

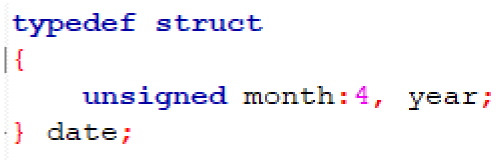
1. INTRODUCTION

This report is made to explain the code the team has made to create the backend part of a bank system in the programming language C.

2. STRUCTS

2.1. Date Structure

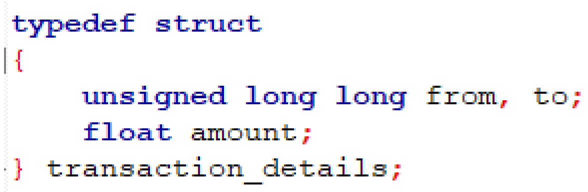
This structure has 2 members of type unsigned int, `month` and `year`. The member `month` is used to store the month when the account was opened, and the `year` member is used to store the year when the account was opened. It is known that months can only be from 1 to 12 so all we need to store the whole range of months is just 4 bits, thus saving memory.



```
typedef struct
{
    unsigned month:4, year;
} date;
```

2.2. Transaction Details Structure

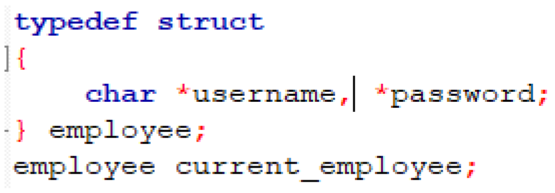
This structure has 3 members, `from`, `to` and `amount`. The `from` and `to` members are of the type `unsigned long long` and are used to store the sender and receivers account numbers, respectively. The `amount` member is of type `float`, and it is used to store the amount of money that was sent/received.



```
typedef struct
{
    unsigned long long from, to;
    float amount;
} transaction_details;
```

2.3. Employee Structure

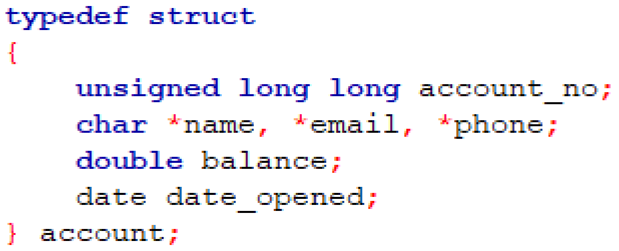
This structure has 2 members of type `string`, `username` and `password`, which store the username and password of an employee.



```
typedef struct
{
    char *username, *password;
} employee;
employee current_employee;
```

2.4. Account Structure

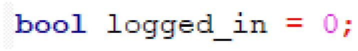
This struct has 6 members, `account_no`, `name`, `email`, `phone`, `balance` and `date_opened`. The `account_no` member is of the type `unsigned long long` and it stores the account number. The `name, email` and `phone` members are of the type `string,` and they store the name, e-mail, and phone number (phone numbers must be stored in a string as they may begin with a 0 and are not used in any calculations), respectively. The `balance` member is of type `double` and is used to store the amount of money in the account. The `date_opened` member is an instance of the struct `date` and it stores when the account was opened.



```
typedef struct
{
    unsigned long long account_no;
    char *name, *email, *phone;
    double balance;
    date date_opened;
} account;
```

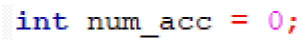
3. GLOBAL VARIABLES & ARRAYS

- The variable `logged_in` is a `boolean` variable and signifies whether there is a user currently logged in the program or not. It is initially equal to 0/`false`.



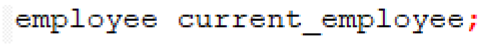
```
bool logged_in = 0;
```

- The variable `num_acc` is of the type `int` and it represents the number of accounts.



```
int num_acc = 0;
```

- `current_employee` is an instance of the `employee` struct and it is used to store the name and password of the employee that is currently logged in.



```
employee current_employee;
```

- `accounts` is an array of pointers to instances of the `account` structure.

```
account** accounts = NULL;
```

4. SUPPLEMENTARY FUNCTIONS

4.1. Read Integer Function

Description:

The `readInteger` function is a validation function that takes a string from the user and returns the input to the calling function if the input passed all checks, otherwise an error message is returned to indicate that there is a problem in the input text.

Input:

The user is asked to enter a number.

Mechanism:

- The function reads a string from the user.
- If the first character is a '\n' the function returns -1
- It validates the entered string:
    - Uses the `isdigit()` built-in function to check if anything other than digits is present.
- If there is any character that is not a digit, then the function returns -1
- The string is converted into an integer and then returned to the calling function

4.2. Read Account Number Function

Description:

The `read_account_no` function is a validation function that takes a string from the user and returns the input to the calling function if the input passed all checks, otherwise an error message is returned to indicate that there is a problem in the input text.

Input:

The user is asked to enter an account number.

Mechanism:

- The function reads a string from the user.
- It validates the entered string:
    - Uses the `isdigit()` built-in function to check if anything other than digits is present.
    - Checks the length of the entered string to be exactly 10 digits.
- If either of the errors exists, the function returns 0.
- If the input is -1:
    - The function returns 1.
    - This is used by the calling function to indicate that the user needs to return to the main menu.

4.3. Contain Special Characters Function

Description:

The `cont_spec` function is a boolean function that takes a string and an integer called `mode` and returns whether it has any special character according to the mode.

Input:

The function receives a string and a mode as a parameter from a function and then returns a boolean to it.

Mechanism:

- The function receives the string and the mode from the calling function.
- If mode is equal to 1, the function returns 1 if any special character is found using the built-in function is `ispunct`.
- If mode is equal to 2, the function returns 1 if the special characters that are not allowed in an e-mail are found.

- If none of these special characters are found, then the function returns 0.

4.4. Contain Digit Function

Description:

The `cont_dig` function is a validation function that takes a string and loops until its end (\0) checking whether the string contains a digit or not.

Input:

It receives a string (character pointer).

Mechanism:

- The function receives a string from the calling function.
- A For loop is executed
  - o It checks on each character whether it is a digit or not.
  - o If at least one digit is present, the function returns 1.
  - o The loop is terminated when the string is finished (\0)
  - o If the loop is terminated, and the function did not return 1, it will return 0, indicating that the string does not contain any digits.

4.5. Read Amount Function

Description:

The `readAmount` function is a validation function that takes a string from the user and returns the input to the calling function if the input passed all checks, otherwise an error message is returned to indicate that there is a problem in the input text.

Input:

The user is asked to enter a number.

Mechanism:

- The function reads a string from the user.

- If the first character is a '\n' the function returns -1

- Then we loop on the input string
    - If the character is a '.' we increment a counter and if it exceeds 1 the function returns -1
    - If the first character is '-' then we raise a flag called `neg` and then return -2.
    - If the character is not a digit the function returns -1
- After we have checked that the string doesn't have any errors, we will loop on the string again to get each part of the number
    - First, we loop till before the '.' to get the number before the decimal point and store it in a variable called `num`
    - Then we loop starting after the '.' to get the number before the decimal point and store it in a variable called `dec` and we also get the number of digits after the '.' And store it in `post`
- To get the result we divide `dec` by 10^{post} and then add it to `num`
- Lastly, the function returns the result to the calling function

4.6. Read Name Function

Description:

The `readName` function is a validation function that takes a string from the user and returns the input to the calling function if the input passed all checks, otherwise an error message is returned to indicate that there is a problem in the input text.

Input:

The user is asked to enter a name.

Mechanism:

- The function reads a string from the user.
- It validates the entered string:
    - Checks that the string doesn't start with an enter (\n) or a space
    - Uses the `cont_dig(), cont_spec()` functions to check if anything other than alphabets is present.
- If the string starts with an enter (\n), the function returns 1.
- If the string starts with a space, the function returns 2.

- If the string contains digits or special characters, the function returns 3.

4.7. Read Email Function

Description:

This function is made to check the entered email is in correct form.

Input:

The user is asked to enter an email.

Errors:

- Email contains characters other than:’+’ ‘-’ ‘_’ ‘@’ ‘.’
- Email contains more than 1 ‘@’, or not at all.
- Email doesn’t contain at least 1 ‘.’

Mechanism:

- The function reads a string from the user and returns the input to the calling function if the input passes, then the function returns the string itself.
- If the string is not in correct form, then the string “NULL” is returned.
- Here, the function compares each character using a loop on the string.

4.8. Read Phone Function

Description:

It is a validation function that takes a string from the user containing the phone number and sees if it’s in correct format.

Input:

The user is asked to enter a phone number.

Errors:

- Phone number contains plus sign anywhere but in the beginning.
- Phone number includes a character.

Mechanism:

- The function reads a string from the user and returns the input to the calling function if the input passes, then the function returns the string itself.
- If the string is not in correct form, then the string “NULL” is returned.
- here, a predefined function **isdigit** (which returns 1 when a digit is found) to check on each character in the string using a loop.

4.9. Binary Search Function

Description:

The `binary_search` function is a searching function that takes an account number and address of the account index from the user and returns “found” integer which indicated if the account was found or not, with its respective account index.

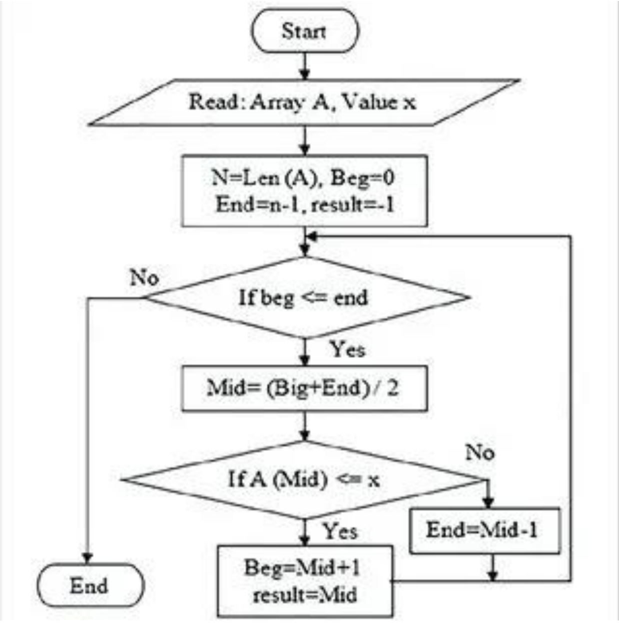
Input:

The user is asked to enter an account number.

Mechanism:

- The function reads the account number and address of account index from the user.
- Four integers are defined:
    - low, which is assigned to 0
    - high, which is assigned to the number of accounts – 1
    - *mid, which receives the address of the account index
    - found, which is assigned to 0
- A While loop is executed
    - *mid is assigned to average of high and low
    - If account number = account number at index (*mid) in the array of accounts structs, found is assigned to 1
    - Else if account number < account number at index (*mid) in the array of accounts structs, high is assigned to *mid - 1
    - Else low is assigned to *mid + 1
    - The loop is terminated when one of two conditions is satisfied:
        - found = 1, which happens when the account is found

    - low > high, which happens when the array of pointers to account structs is finished



4.10. Print Account Function

Description:

The `printAccount` function is a printing function that takes a pointer to an account struct and prints all the details of this account.

Input:

The function is called and the pointer to an account struct is passed as an argument to it.

Mechanism:

- The function prints all the data of the given account with its respective titles as needed.

4.11. Create Transaction Files Function

Description:

The `create_transaction_file` function generates a transaction file associated with a specified account number, ensuring the file exists for recording transaction data.

Input:

- ***unsigned long long account_no:*** The account number for which the transaction file is created.

Mechanism:

1. ***File Name Generation:***
   - Generates the file name based on the account number (filename) using `snprintf` to ensure a consistent format **("<account_no>.txt").**
2. ***File Opening:***
   - Attempts to open the file associated with the account number in read mode **("r").**
   - If the file is not found, it creates a new file with write mode **("w").**
   - Closes the file after opening to ensure file creation or to handle any potential errors during the opening process.

4.12. Save Transaction Function

Description:

The `saveTransaction` function appends transaction data (from, to, and amount) to a file associated with a specified account number (**“account_no.txt”**), handling errors such as file not found or unsuccessful data saving.

Input:

- ***unsigned long long account_no****:* The account number for which the transaction data is to be saved.
- ***unsigned long long from****:* The source of the transaction.

- ***unsigned long long to:*** The destination of the transaction.
- ***double amount:*** The amount involved in the transaction.

Errors:

File Not Found:

- If the function is unable to open the transaction file associated with the given account number (filename), it prints an error message ("**Error: <filename> file not found"**) and returns without saving the transaction data.

Data Saving Error:

- If there is an error while attempting to save the transaction data to the file, it prints an error message (**"Error: data could not be saved"**).

Mechanism:

1. ***File Name Generation:***
   - Generates the file name based on the account number (`filename`) using `snprintf` to ensure a consistent format (**"<account_no>.txt"**).
2. ***File Opening:***
   - Attempts to open the file associated with the account number in read mode (**"r"**).
   - If the file is not found, prints an error message and returns without saving the transaction data.
   - If file is found, reopens the file in append mode (**"a"**) for adding new transaction data.
3. ***Data Saving:***
   - Writes the transaction data (from, to, and amount) to the file using `fprintf`.
   - Checks if the data saving operation was successful; if not, prints an error message. (**"Error: data could not be saved"**)
4. ***File Closure:***

- Closes the file after saving the transaction data `(fclose(fp))`.

4.13. Decode Text Function

Description:

The `decodeText` function takes a text representation of an account, separates the information into each field, and creates an account structure with the decoded data.

Input:

- ***char* line:*** A string containing the comma-separated values representing an account (account number, name, email, balance, phone, and date opened).

Mechanism:

1. ***File Name Generation:***
   - Generates the file name based on the account number (filename) using `snprintf` to ensure a consistent format **("<account_no>.txt").**
2. ***Tokenization:***
   - Utilizes `strtok` to tokenize the input string (line) based on commas, extracting individual fields.
3. ***Decoding Fields:***
   - Extracts the account number (`accNum`) as an unsigned long long using `strtoull`.
   - Retrieves the name, email, balance, phone, and date opened fields using successive calls to `strtok` and appropriate conversion functions (`strtod` for balance, `atoi` for integers).
4. ***Structure Creation:***
   - Creates a date structure (`date_opened`) with the month and year components.
   - Calls the `constAcc` function to construct and return an account structure with the decoded information.

4.14. Sorting Comparison Functions

Description:

The sorting comparison functions (`SortByNum`, `SortByName`, `SortByBalance`, and `SortByDate`) define the criteria for ordering elements within an array of pointers to `account` structures, facilitating sorting operations using the `qsort` function. (ascending order)

Input:

- **const void *a**: A pointer to the first element to be compared.
- **const void *b**: A pointer to the second element to be compared.

Mechanism:

1. ***SortByNum:***
    - Compares account numbers and returns 1 if the account number of the first element is greater, -1 if it is smaller, and 0 if they are equal.
2. ***SortByName:***
    - Uses `strcmp` to compare the names of two accounts alphabetically.
3. ***SortByBalance:***
    - Compares balances and returns 1 if the balance of the first element is greater, -1 if it is smaller, and 0 if they are equal.
4. ***SortByDate:***
    - Compares dates, first based on the year and then on the month if the years are equal.
    - Returns 1 if the date of the first element is later, -1 if it is earlier, and 0 if they are equal.

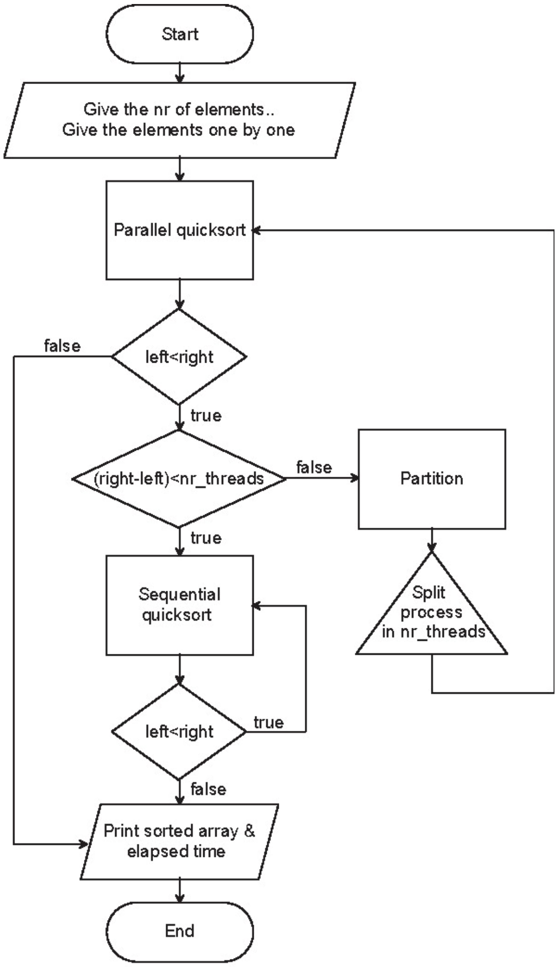


Fig. 4. Parallel quicksort flowchart diagram

4.15. Construct Account Function

Description:

The `constAcc` function dynamically allocates memory for an `account` structure, initializes its fields with the provided data, and creates a transaction file associated with the account number.

Input:

- **unsigned long long account_no:** The account number.
- **char *name:** The account holder's name.
- **char *email:** The account holder's email.
- **double balance:** The initial account balance.
- **char *phone:** The account holder's phone number.
- **date date_opened:** The date the account was opened.

Mechanism:

1. ***Memory Allocation:***
   - Allocates memory for an account structure using `malloc`.
   - Allocates memory for the name, email, and phone fields within the account structure, copying the corresponding data.
2. ***Field Initialization:***
   - Initializes the fields of the account structure with the provided data (account number, name, email, balance, phone, and date opened).
3. ***Transaction File Creation:***
   - Calls the `create_transaction_file` function to create a transaction file associated with the account number.
4. ***Return:***
   - Returns a pointer to the dynamically allocated and initialized account structure.

Note: opposite function is called `distAcc` which does the opposite thing by deallocating the fields and freeing the account pointer.

5. MAIN FUNCTIONS

5.1. Void login()

Description:

The `login` function is responsible for taking the username and password from the user to verify his/her credentials to securely access the accounts data.

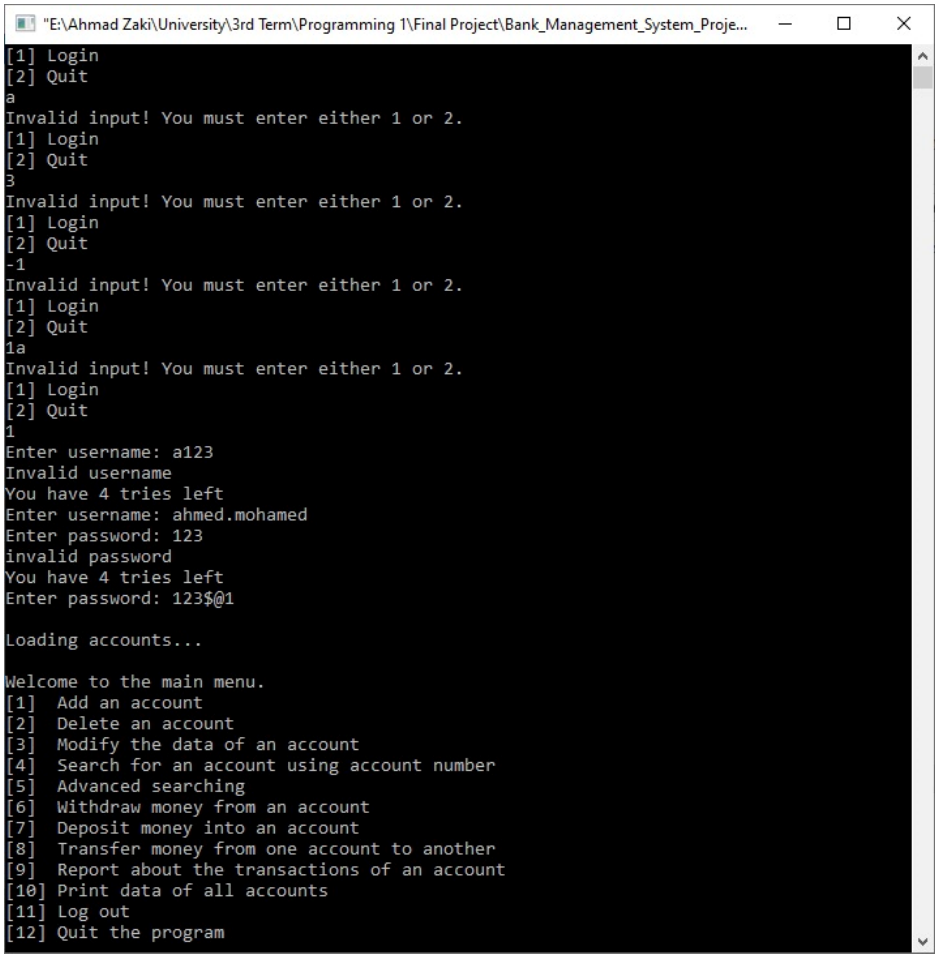
Input:

Here the user is asked to enter an existing username and a password matching to that username that’s included in the users.text file.

Mechanism:

- First of all, the users.text file is used, and a pointer is assigned to it to read from that file.
- Next the user is asked to enter the username.
- Using `strtok`, the data written is separated to read username and password separately.
- Next, a is set to compare the entered username to each username in the file.
- If username entered is found to be correct, then program compares the password next to it.
- Error message is presented if one of the two are incorrect and user is asked to enter again
- User has only 5 tries, then system automatically logs out.

Test sample:


"E:\Ahmad Zaki\University\3rd Term\Programming 1\Final Project\Bank_Management_System_Proje...

```
[1] Login
[2] Quit
a
Invalid input! You must enter either 1 or 2.
[1] Login
[2] Quit
3
Invalid input! You must enter either 1 or 2.
[1] Login
[2] Quit
-1
Invalid input! You must enter either 1 or 2.
[1] Login
[2] Quit
1a
Invalid input! You must enter either 1 or 2.
[1] Login
[2] Quit
1
Enter username: a123
Invalid username
You have 4 tries left
Enter username: ahmed.mohamed
Enter password: 123
invalid password
You have 4 tries left
Enter password: 123$@1

Loading accounts...

Welcome to the main menu.
[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
```

5.2. Void load()

Description:

The `load` function is responsible for reading account information from a file named **"accounts.txt"**, counting the number of accounts, dynamically allocating memory for an array of pointers to `account` structures, decoding the text representation of each account using `decodeText` function, and populating the array.

Input:

No input is required in this function.

Errors:

No File Found

- accounts.txt file must be found in the folder. Else, an error message is printed **“Error: accounts.txt file not found.”**

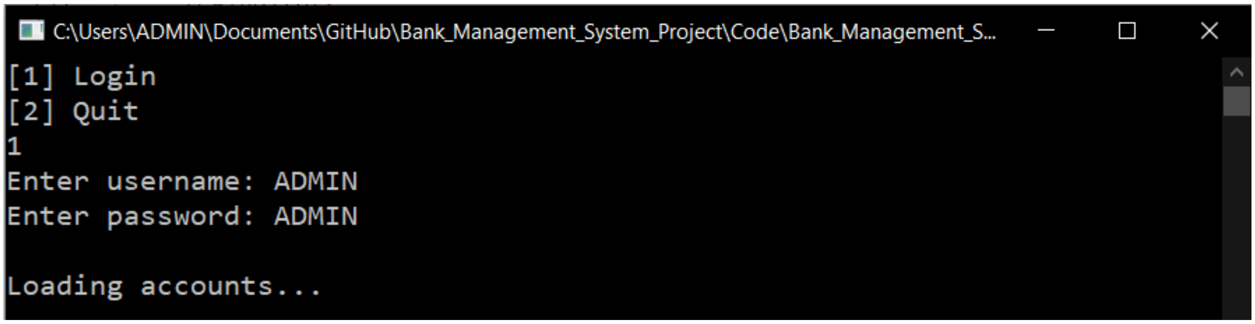
Algorithm:

1. ***File Opening:***
   - Opens the **"accounts.txt"** file in read-only mode (**"r"**).
   - If the file cannot be opened (returns `NULL`), it prints an error message, and the program exits.
2. ***Counting Accounts:***
   - Initializes a character array `record` to store each line read from the file.
   - Uses `fgets` to read each line from the file.
   - If the line is empty (contains only a newline character **‘\n’**), it skips it.
   - Increments the global `num_acc` variable for each non-empty line, representing an account.
3. ***Memory Allocation:***

- Allocates memory for a global array of pointers to `account` structures called `accounts` using `malloc`. The size of the allocation is based on the count of accounts (`num_acc`).

4. ***Decoding and Populating Accounts:***
   - Resets the file pointer to the beginning of the file using `rewind(fp).`
   - Uses a loop to iterate (`num_acc`) times over each line in the file.
   - Reads each line using `fgets`.
   - If the line is empty, it skips it.
   - Decodes the text of the line into an `account` structure using the `decodeText` function and returns a pointer to `account`.
   - Stores the returned decoded pointer to `account` structure in the array of `accounts`.

5. ***File Closure:***
   - Closes the file after processing to free up system resources using `fclose(fp).`


C:\Users\ADMIN\Documents\GitHub\Bank_Management_System_Project\Code\Bank_Management_S...

```
[1] Login
[2] Quit
1
Enter username: ADMIN
Enter password: ADMIN

Loading accounts...
```

Note: opposite function is called `unload` which does the opposite thing by distructing accounts in array and then freeing the accounts array.

5.3. Void query_search()

Description: The `query_search` function is used to search for an account using Account Number.

Input: The user enters the Account Number.

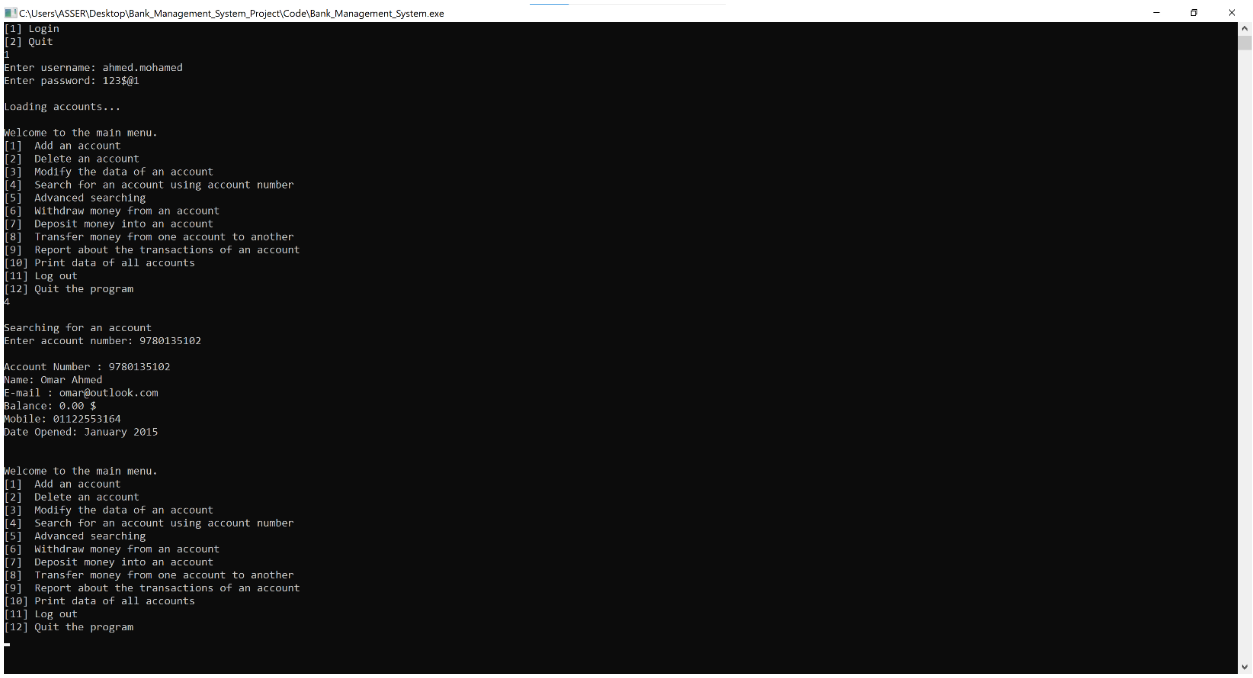
Output: All details related to the account number is printed, such as (Name, E-mail Address, Balance, Mobile, Date Opened).

Errors:

- Account Number must contain only digits (No alphabets or special characters).
- Account Number must be made of 10 digits, no more or less.
- Account Number must be found in accounts.txt file.

Mechanism:

- The user is prompted to enter the Account Number.
- The array of pointers of structs is sorted ascendingly according to the Account Number using the `qsort` pre-defined function in C.
- The entered Account Number is validated using the `read_account_no` function.
- The Account Number is searched for using the `binary_search` function.
- If not found, an error message is printed indicating that the account was not found.
- If found, all details related to the account number are printed.
- The user is then returned to the Main Menu.



C:\Users\ASSER\Desktop\Bank_Management_System_Project\Code\Bank_Management_System.exe

```
[1] Login
[2] Quit
1
Enter username: ahmed.mohamed
Enter password: 123$@1

Loading accounts...

Welcome to the main menu.
[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
4

Searching for an account
Enter account number: 9780135102

Account Number : 9780135102
Name: Omar Ahmed
E-mail : omar@outlook.com
Balance: 0.00 $
Mobile: 01122553164
Date Opened: January 2015


Welcome to the main menu.
[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
```

5.4. Void advanced_search()

Description:

The `advanced_search` function is responsible for taking a keyword from the user and searching for this keyword in all users' data and print all accounts with matching results.

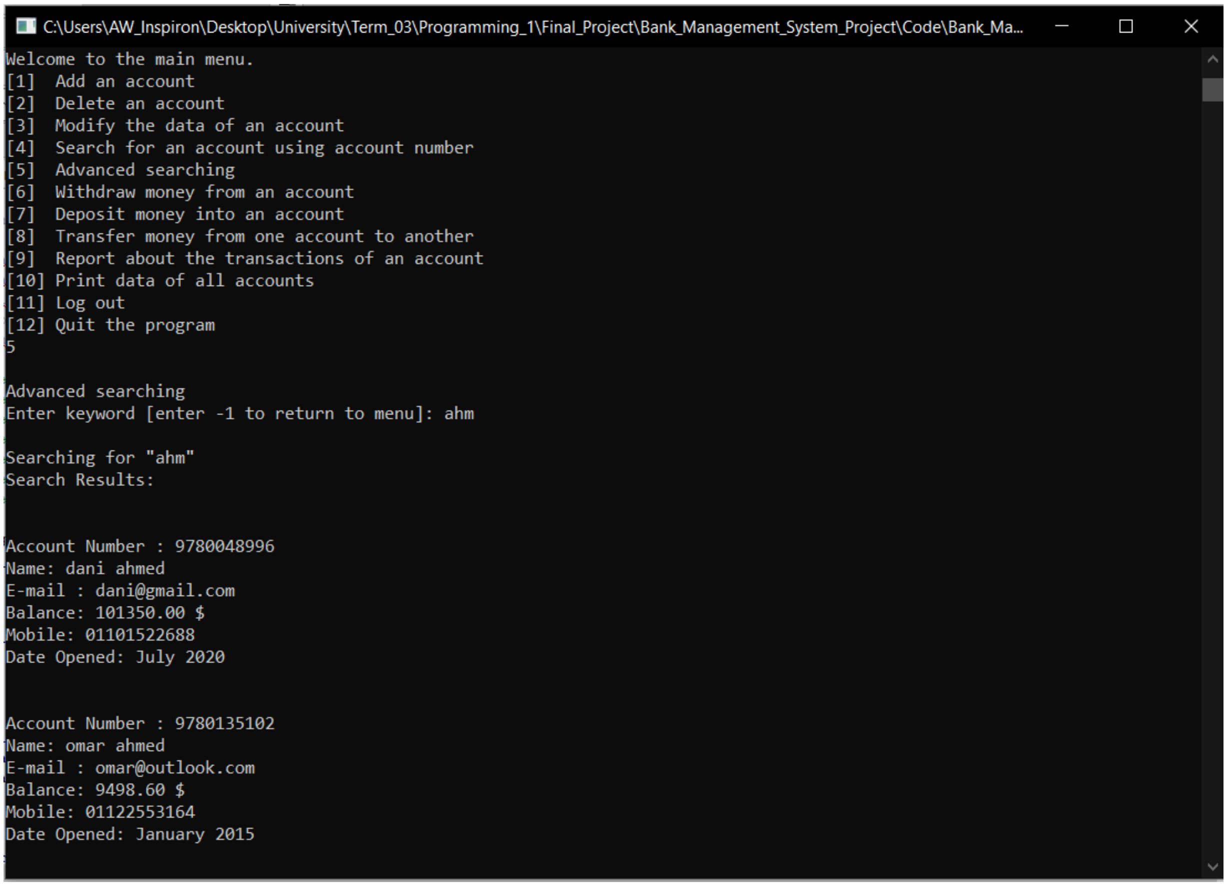
Input:

The user can enter any type of data as an input with only limitation to the starting character that it should only be a character or a number.

Mechanism:

- This function is based on linear search.
- It compares the entered keyword with every account record available.
- Multiple matches to the supplied keyword are possible.
- For more efficiency, the input taken from the user is checked:
  - If it contains only numbers, the search is conducted only on account number and phone number.
  - If it contains other characters, the search is conducted on name and email address only.
- This differentiation approximately halves the comparisons needed.
- If the search is conducted on name and email address, the text is first processed to be all lowercase, this ensures correct string matching.

Test sample:



C:\Users\AW_Inspiron\Desktop\University\Term_03\Programming_1\Final_Project\Bank_Management_System_Project\Code\Bank_Ma...

```
Welcome to the main menu.
[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
5

Advanced searching
Enter keyword [enter -1 to return to menu]: ahm

Searching for "ahm"
Search Results:


Account Number : 9780048996
Name: dani ahmed
E-mail : dani@gmail.com
Balance: 101350.00 $
Mobile: 01101522688
Date Opened: July 2020


Account Number : 9780135102
Name: omar ahmed
E-mail : omar@outlook.com
Balance: 9498.60 $
Mobile: 01122553164
Date Opened: January 2015
```

5.5. Void add()

Description:

The `add` function is responsible for adding a new non existing account.

Input:

User here is asked to enter: new account number, name of account holder, phone number, email address.

Errors:

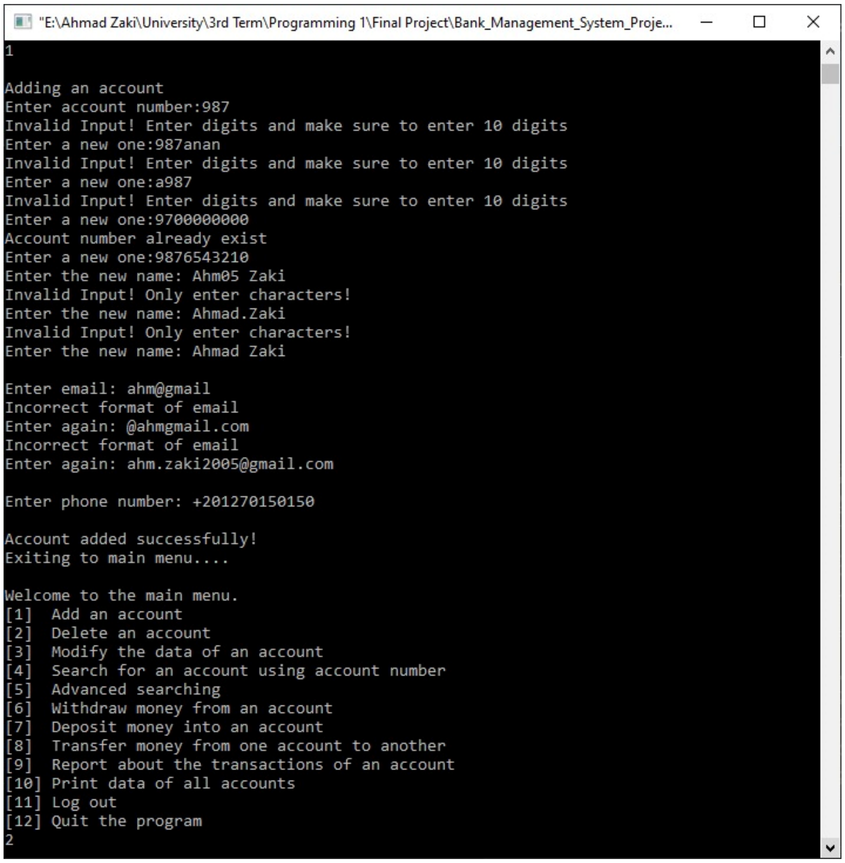
- Account number contains characters.
- Account number larger than 10 digits.
- Name contains digits.
- Phone number contains characters.

- Email address not according to known format (explained more in readEmail() function).

Mechanism:

- First user is asked to enter a non-existing account number & must be 10 digits exactly which is checked by `read_account_no` function.
- Name is checked by `readName()` function which makes sure name is only alphabetic characters.
- Email address is checked by `readEmail` function.
- Phone number is checked using `readPhone` function.
- All above are in a do while loop which doesn't end until user enters each input using suggested format.
- After that, the size of the array of structs is incremented by one using `realloc()` function.
- Finally, the date is automatically set from the system using predefined library stored in C which is `time.h`.
- Then a predefined struct. Is added (`time_t`) which has a variable `currentTime`.
- All entered data is stored inside the new struct using `const_acc` function. (explained in detail later)
- At last, the user is sent back to the main menu to choose the next action he wished to do.

Test sample:


"E:\Ahmad Zaki\University\3rd Term\Programming 1\Final Project\Bank_Management_System_Proje...

```
1

Adding an account
Enter account number:987
Invalid Input! Enter digits and make sure to enter 10 digits
Enter a new one:987anan
Invalid Input! Enter digits and make sure to enter 10 digits
Enter a new one:a987
Invalid Input! Enter digits and make sure to enter 10 digits
Enter a new one:9700000000
Account number already exist
Enter a new one:9876543210
Enter the new name: Ahm05 Zaki
Invalid Input! Only enter characters!
Enter the new name: Ahmad.Zaki
Invalid Input! Only enter characters!
Enter the new name: Ahmad Zaki

Enter email: ahm@gmail
Incorrect format of email
Enter again: @ahmgmail.com
Incorrect format of email
Enter again: ahm.zaki2005@gmail.com

Enter phone number: +201270150150

Account added successfully!
Exiting to main menu....

Welcome to the main menu.
[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
2
```

5.6. Void delete_account()

Description:

The `delete_account` function is responsible for deleting an account from the array of pointers to structs “accounts” and delete its respective transactions file.

Input:

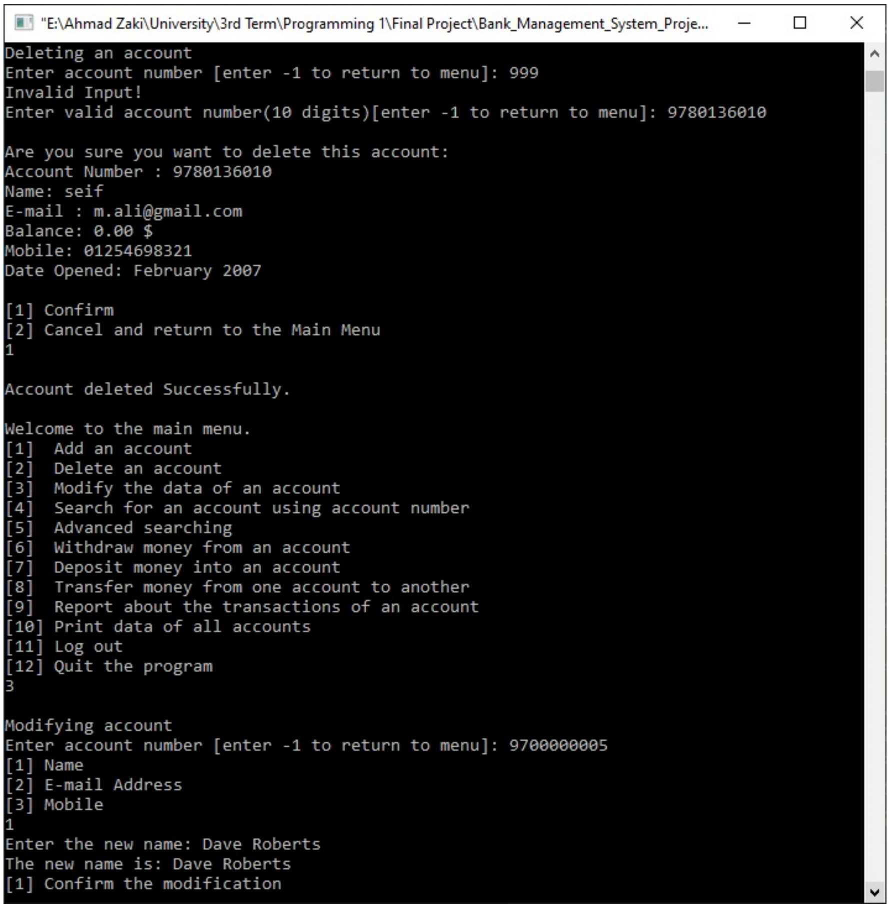
The user is asked to enter an account number.

Mechanism:

- A linear search is performed on the array of accounts (structs).
- If the account is found:
    - The balance of this account is checked.
    - If a positive balance is found:

    - An error message is prompted to the user.
    - The function returns to the menu.
  - If the account balance is equal to zero:
    - The account details are printed to the user.
    - A confirmation message appears to the user.
    - If confirmed:
      - The account struct is freed from memory.
      - All structs are shifted backward.
      - The transactions file of this account is deleted.

Test sample:


"E:\Ahmad Zaki\University\3rd Term\Programming 1\Final Project\Bank_Management_System_Proje...

```
Deleting an account
Enter account number [enter -1 to return to menu]: 999
Invalid Input!
Enter valid account number(10 digits)[enter -1 to return to menu]: 9780136010

Are you sure you want to delete this account:
Account Number : 9780136010
Name: seif
E-mail : m.ali@gmail.com
Balance: 0.00 $
Mobile: 01254698321
Date Opened: February 2007

[1] Confirm
[2] Cancel and return to the Main Menu
1

Account deleted Successfully.

Welcome to the main menu.
[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
3

Modifying account
Enter account number [enter -1 to return to menu]: 9700000005
[1] Name
[2] E-mail Address
[3] Mobile
1
Enter the new name: Dave Roberts
The new name is: Dave Roberts
[1] Confirm the modification
```

5.7. Void modify_acc()

Description: The `modify_acc` function is used to modify account details such as (Name, E-mail Address, and Mobile) field by field.

Input: The user enters the Account Number.

Output: The details related to the Account Number is modified in accounts.txt file as requested by the user.

Errors:

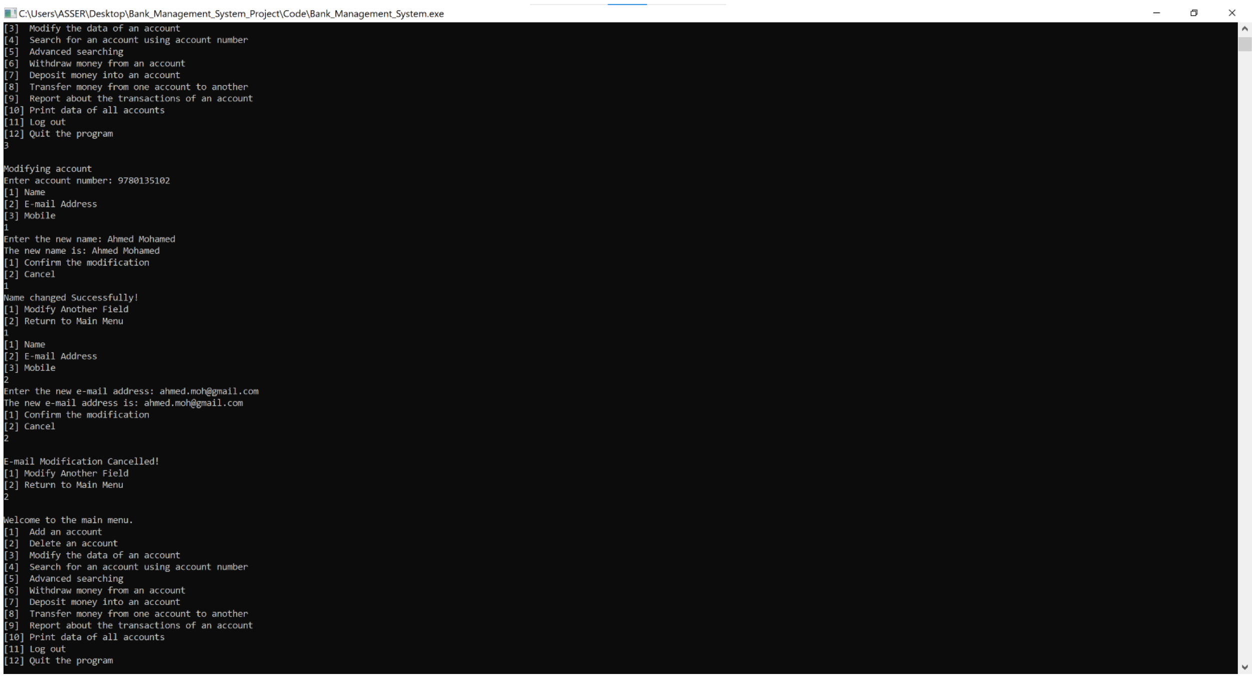
- Account Number must contain only digits (No alphabets or special characters).
- Account Number must be made of 10 digits, no more or less.
- Account Number must be found in accounts.txt file.

Mechanism:

- The user is prompted to enter the Account Number.
- The array of pointers of structs is sorted ascendingly according to the Account Number using `qsort` pre-defined function in C.
- The entered Account Number is validated using the `read_account_no` function.
- The Account Number is searched for using `binary_search` function.
- If not found, an error message is printed indicating that the account was not found.
- If found, the user is prompted to choose a field to modify:
  - Option 1: Modify name
  - Option 2: Modify e-mail address
  - Option 3: Modify mobile
- Input is validated by the `readInteger` function.
- If modifying the name field option is chosen:
- A message to enter the name is printed, and the name is validated using the `readName` function.
- If modifying the e-mail address option is chosen:
- A message to enter the e-mail address is printed, and the e-mail address is validated using the `readEmail` function.
- If modifying the mobile option is chosen:
- A message to enter the mobile number is printed, and the mobile number is validated using the `readPhone` function.

- In each field, the modification is printed out with a confirmation message and two options:
  - Option 1: Confirm the modification
  - Option 2: Cancel
- After that, the user is given another two options:
  - Option 1: Modify another field
  - Option 2: Return to Main Menu

Test sample:



C:\Users\ASSER\Desktop\Bank_Management_System_Project\Code\Bank_Management_System.exe

```
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
3

Modifying account
Enter account number: 9780135102
[1] Name
[2] E-mail Address
[3] Mobile
1
Enter the new name: Ahmed Mohamed
The new name is: Ahmed Mohamed
[1] Confirm the modification
[2] Cancel
1
Name changed Successfully!
[1] Modify Another Field
[2] Return to Main Menu
1
[1] Name
[2] E-mail Address
[3] Mobile
2
Enter the new e-mail address: ahmed.moh@gmail.com
The new e-mail address is: ahmed.moh@gmail.com
[1] Confirm the modification
[2] Cancel
2

E-mail Modification Cancelled!
[1] Modify Another Field
[2] Return to Main Menu
2

Welcome to the main menu.
[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
```

5.8. Void withdraw()

Description:

The `withdraw` function is responsible for withdrawing money from an account.

Input:

User here is asked to input the account number that that’s wanted to be withdraw from, and the amount that’s required to be drawn from it.

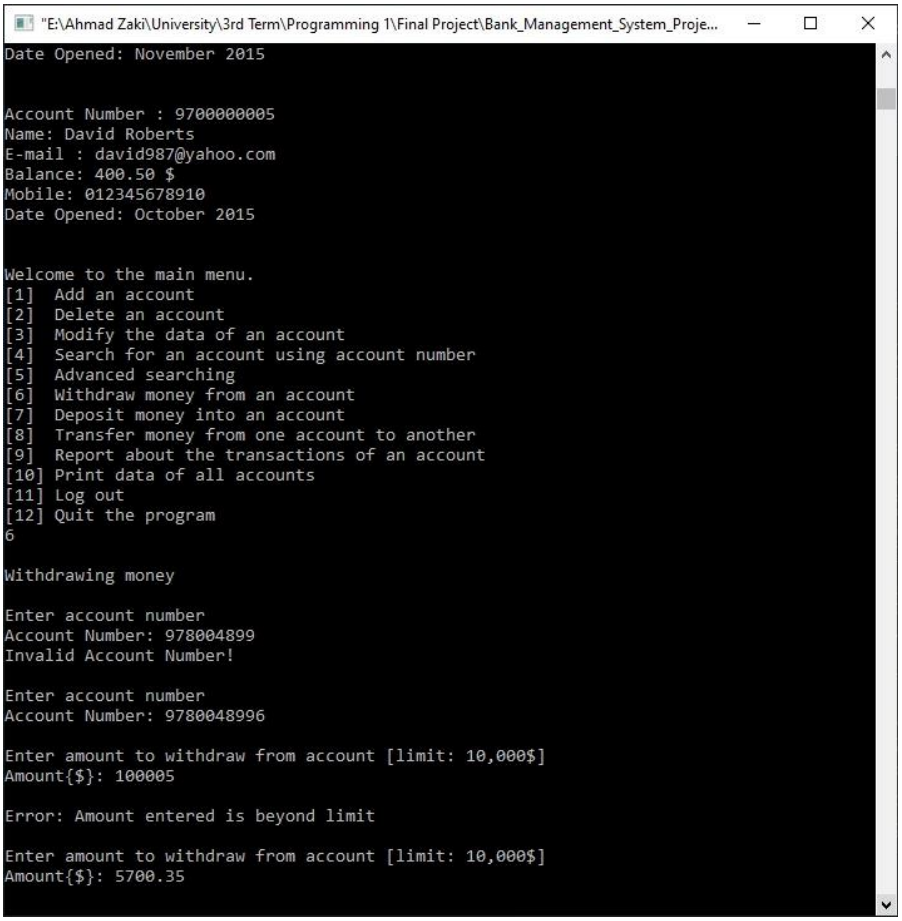
Errors:

- Entering a non-existent account number
- Entering an account number not in correct format as instructed.
- Entering amount to be withdrawn exceeding the limit{$10,000}
- Entering the amount in an incorrect format
- Entering an amount exceeding the balance of chosen account number.

Mechanism:

- Firstly the user is asked to enter an account number which is checked upon using the `read_account_no()` function to make sure the correct format is entered.
- A do while loop is used and not exited until account number is entered using correct format.
- Next, `qsort` function is used to sort the accounts according to their account numbers.
- The entered account number is then compared to each account number inside the array of struct, using the `binary_search()` function.
- If account not found, an error message is printed then the `withdraw` function is called again to enter a new account number.
- The function will stop calling itself only if account number exists.
- User then is asked to enter an amount to be withdrawn from an account.
- Then a set of if statements is used inside a do while loop to validate the amount entered.
- The loop ends only if: the amount is only digits, amount is not a negative number(checked using readAmount),amount is not beyond the $10,000 limit and that it doesn't exceed the balance of the account chosen.
- If one of these errors is present, then an error message is presented according to the error present.
- A confirmation message is printed to confirm the transaction.
- After that the balance is corrected inside the struct by subtracting the amount entered.
- Next the all the transaction details are sent to `saveTransaction()` function (will be explained later) so that it would be saved.
- Finally, the `save()` function is called to save all the actions done.

Test sample:


"E:\Ahmad Zaki\University\3rd Term\Programming 1\Final Project\Bank_Management_System_Proje...

```
Date Opened: November 2015


Account Number : 9700000005
Name: David Roberts
E-mail : david987@yahoo.com
Balance: 400.50 $
Mobile: 012345678910
Date Opened: October 2015


Welcome to the main menu.
[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
6

Withdrawing money

Enter account number
Account Number: 978004899
Invalid Account Number!

Enter account number
Account Number: 9780048996

Enter amount to withdraw from account [limit: 10,000$]
Amount{$}: 100005

Error: Amount entered is beyond limit

Enter amount to withdraw from account [limit: 10,000$]
Amount{$}: 5700.35
```

5.9. Void deposit()

Description:

The `deposit` function prompts the user to enter an account number and a deposit amount, validates inputs, confirms the transaction, updates the account balance, and records the transaction details in the account's file.

Input:

No input is required in this function.

Errors:

Invalid Account Number:

- If the entered account number is not a 10-digit positive number, the `read_account_no` function returns 0.

- If 0 is returned, it prints an error message (**"Invalid Account Number!"**) and prompts the user to re-enter a valid account number.
- This keeps on looping until a valid account number is entered.

Account Not Found:

- If the specified account number is not found in the array, it prints an error message (**"The Specified Account is not found!"**), recursively calls itself, and prompts the user to enter the account number again.

Invalid Amount:

- Handles various scenarios related to invalid input amounts, such as negative amounts, non-numeric input, amounts of 0, and amounts exceeding the limit of $10,000. It prints appropriate error messages for each case and prompts the user to enter a valid amount.
- The `readAmount` function returns different values according to each case and an error message is printed for each corresponding amount value:
    - Returns -2 (negative amounts)
      "Error: Amount entered cannot be negative"
    - Returns -1 (characters entered)
      "Error: Invalid Amount (Do not enter characters)"
    - Returns 0
      "Error: Amount entered cannot be 0 $"
    - If amount > 10,000
      "Error: Amount entered is beyond limit\n"

Mechanism:

1. ***Account Number Entry:***

- Prompts the user to enter an account number using the `read_account_no` function.
- Validates the entered account number and prompts for re-entry if invalid.

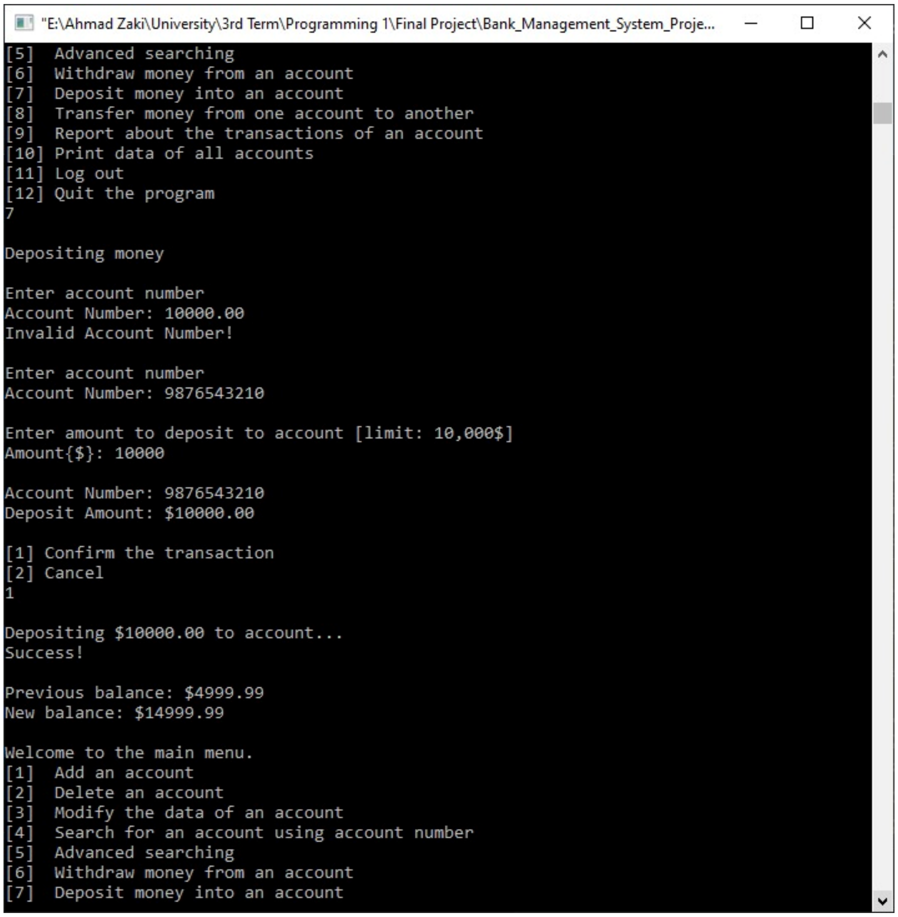
2. ***Account Sorting and Binary Search:***

- Utilizes `qsort` to sort the array of accounts based on account numbers using the `SortByNum` comparison function.

- Performs a binary search to locate the specified account using `binary_search` function.
- If the account is not found, displays an error message, recursively calls itself for another attempt, and returns.

3. ***Deposit Amount Entry:***
   - Prompts the user to enter a deposit amount within the range of $0.01 to $10,000 using the `readAmount` function.
   - Validates the amount, checking for negative values, non-numeric inputs, zero amounts, and amounts exceeding the limit. Prompts for re-entry until a valid amount is provided.
4. ***Transaction Confirmation:***
   - Presents the user with the option to confirm or cancel the transaction.
   - Validates the user's input, prompting for re-entry if an invalid choice is made.
5. ***Transaction Processing:***
   - If the user confirms the transaction, updates the account balance, prints transaction details, and saves the transaction (`saveTransaction`).
   - If the user cancels the transaction, prints a cancellation message and returns. (**"Transaction Cancelled!"**)
6. ***User Confirmation Input Handling:***
   - Continues prompting for confirmation until a valid input (1 for confirmation, 2 for cancellation) is received.

Test sample:



"E:\Ahmad Zaki\University\3rd Term\Programming 1\Final Project\Bank_Management_System_Proje...

```
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
7

Depositing money

Enter account number
Account Number: 10000.00
Invalid Account Number!

Enter account number
Account Number: 9876543210

Enter amount to deposit to account [limit: 10,000$]
Amount{$}: 10000

Account Number: 9876543210
Deposit Amount: $10000.00

[1] Confirm the transaction
[2] Cancel
1

Depositing $10000.00 to account...
Success!

Previous balance: $4999.99
New balance: $14999.99

Welcome to the main menu.
[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
```

5.10. Void transfer()

Description: The `transfer` Function is used to transfer money from one bank account to another.

Input: The user enters the Account Numbers of both the sender and the receiver, and the amount to be transferred.

Output: The sender's Account Number, previous balance, and new balance after transaction is printed out, and same to the receiver.

Errors:

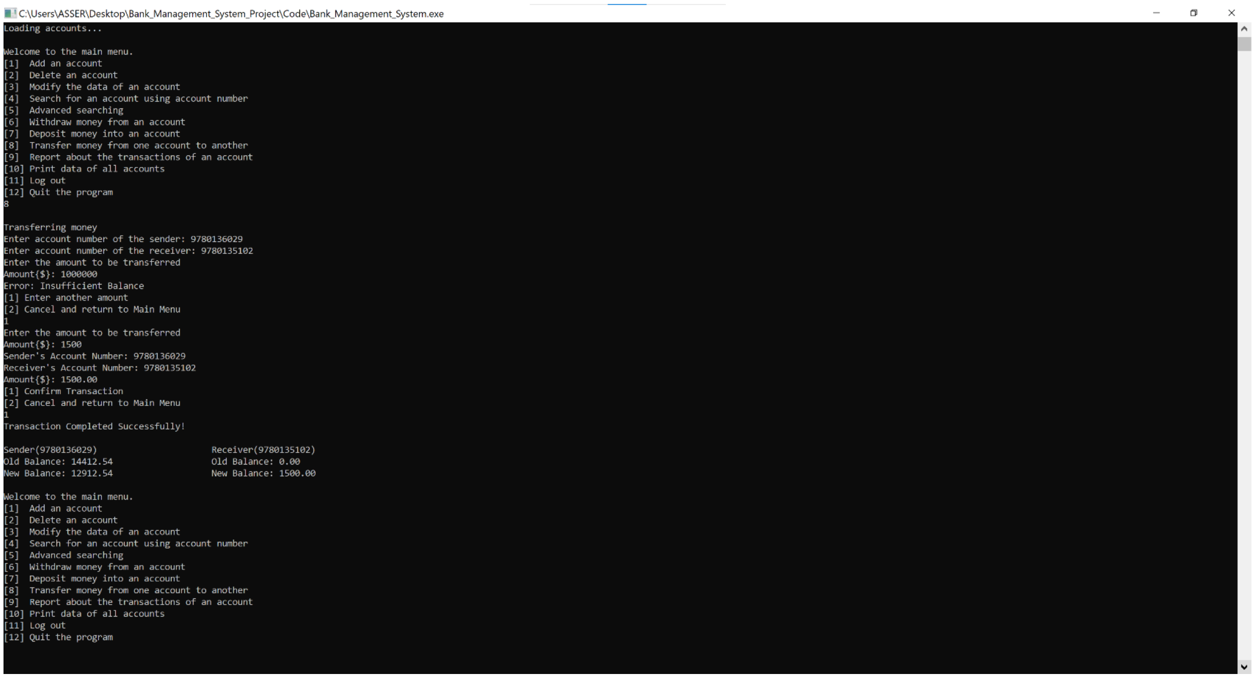
- Account Number must contain only digits (No alphabets or special characters).
- Account Number must be made of 10 digits, no more or less.

- Account Number must be found in accounts.txt file.
- Account Numbers of sender and receiver must be different.
- Amount entered cannot be negative, nor contain characters nor be 0 $.
- Amount entered must be less than the sender’s balance.

Mechanism:

- The user is prompted to enter the Account Number of the sender.
- The array of pointers of structs is sorted ascendingly according to the Account Number using the `qsort` pre-defined function in C.
- The entered Account Number is validated using the `read_account_no` function.
- The Account Number is searched for using the `binary_search` function.
- If not found, an error message is printed indicating that the account was not found.
- Steps (1-5) are repeated for the receiver.
- The user is then prompted to enter the amount to be transferred (the sender must have sufficient balance for the transaction), which is validated by the `readAmount` function.
- After that, the user is given two options:
  - Option 1: Enter another amount.
  - Option 2: Cancel and return to Main Menu
- Input is validated by the `readInteger` function.
- The sender’s and receiver’s account numbers and the amount to be transferred are printed out with a confirmation message.
  - Option 1: Confirm the transaction.
  - Option 2: Cancel and return to Main Menu.
- If confirmed, a message is printed out to inform the user that the transaction was successful, and the sender’s and receiver’s Account Numbers, previous balances, and new balances after the transaction are printed out in a table form.
- The user is then returned to the Main Menu.

Test sample:



C:\Users\ASSER\Desktop\Bank_Management_System_Project\Code\Bank_Management_System.exe

```
Loading accounts...

Welcome to the main menu.
[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
8

Transferring money
Enter account number of the sender: 9780136029
Enter account number of the receiver: 9780135102
Enter the amount to be transferred
Amount{$}: 1000000
Error: Insufficient Balance
[1] Enter another amount
[2] Cancel and return to Main Menu
1
Enter the amount to be transferred
Amount{$}: 1500
Sender's Account Number: 9780136029
Receiver's Account Number: 9780135102
Amount{$}: 1500.00
[1] Confirm Transaction
[2] Cancel and return to Main Menu
1
Transaction Completed Successfully!

Sender(9780136029)                      Receiver(9780135102)
Old Balance: 14412.54                   Old Balance: 0.00
New Balance: 12912.54                   New Balance: 1500.00

Welcome to the main menu.
[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
```

5.11. Void report()

Description:

The function `report` prints the most recent transactions sorted from the most recent to the oldest. If there are more than 5 transactions, only the 5 most recent will be printed.

Input:

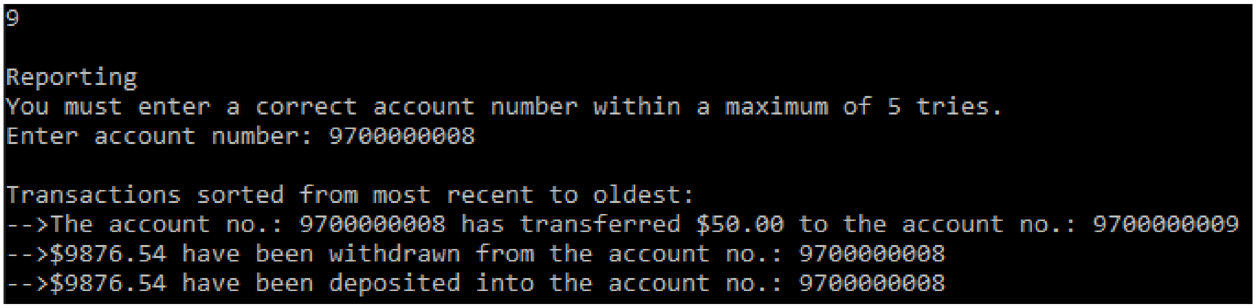
There are two types of inputs. The first type is the account number and it is read from the console. The second type is the transaction details from the acc_no.txt.

Mechanism:

- The function first asks for the account number and calls the function `read_account_no` to read the input.
- The input must be in the valid format within 5 tries, otherwise, the function returns to the function `menu`.
- Then it tries to find the input account number in the accounts array, and if not found, the function returns to the function `menu`.
- We use try to open the acc_no.txt file if it exists.

- If the file doesn't exist, the code exits with exit code 3.
- After that, we initialize an array of `transaction_details` called `transactions`.
- We start taking input from the file acc_no.txt.
- While we have not reached the end of the file:
- The array is shifted by one place towards the end, overwriting the data of the last element completely.
- The first element is then read from the acc_no.txt file.
- Then the array is printed.
- In the array, the symbol '5' represents the client.
- The term "withdraw" means that the account transfers money to the client.
- The term "deposit" is as if the client transfers money to the account.

Test sample:



```
9

Reporting
You must enter a correct account number within a maximum of 5 tries.
Enter account number: 9700000008

Transactions sorted from most recent to oldest:
-->The account no.: 9700000008 has transferred $50.00 to the account no.: 9700000009
-->$9876.54 have been withdrawn from the account no.: 9700000008
-->$9876.54 have been deposited into the account no.: 9700000008
```

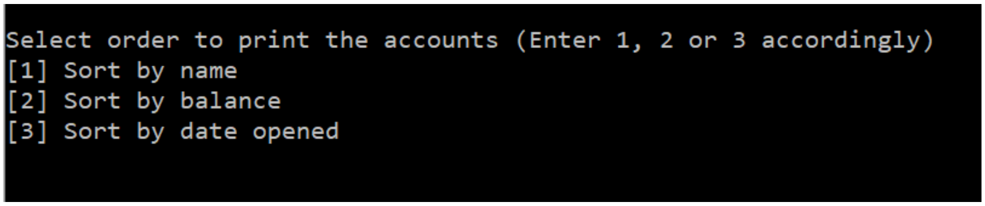
5.12. Void print()

Description:

The `print` function allows the user to select the order (by name, balance, or date opened) in which to print and display the accounts, handling invalid inputs by prompting the user for a valid choice and recursively calling itself.

Input:

User input for selecting the order to print the accounts (sorted by name, balance, or date opened).



```
Select order to print the accounts (Enter 1, 2 or 3 accordingly)
[1] Sort by name
[2] Sort by balance
[3] Sort by date opened
```

Errors:

Invalid Input:

- If the user enters an option other than 1, 2, or 3, it prints an error message (**"INVALID INPUT! Enter 1, 2, or 3 only"**), recursively calls itself, and prompts for a valid input.

Algorithm:

1. ***User Input:***
   - Prompts the user to select the order in which to print the accounts (by name, balance, or date opened) using the `readInteger` function.
2. ***Sorting Accounts:***
   - Uses a switch statement to determine the sorting order based on the user's input (1, 2, or 3).
     - Calls `qsort` to sort the array of accounts accordingly:
     - Case 1: Sorts by name (`SortByName` comparison function).
     - Case 2: Sorts by balance (`SortByBalance` comparison function).
     - Case 3: Sorts by date opened (`SortByDate` comparison function).
     - Default: Prints an error for invalid input and recursively calls itself.
3. ***Printing Accounts:***
   - Iterates through the sorted array of accounts and prints each account using the `printAccount` function.

6.1. Void save()

Description:

The `save` function is responsible for saving any modifications to the accounts array to the accounts.txt file.

Input:

No input is required in this function.

Mechanism:

- This function is called after any transaction performed by the user.
- The accounts.txt file is opened in write mode.
- If accounts file couldn't be opened the code exits with exit code 2.
- The entire file is overwritten with the array of structs.

6.2. Void log_out()

Description:

The function `log_out` is responsible for logging out of the current user.

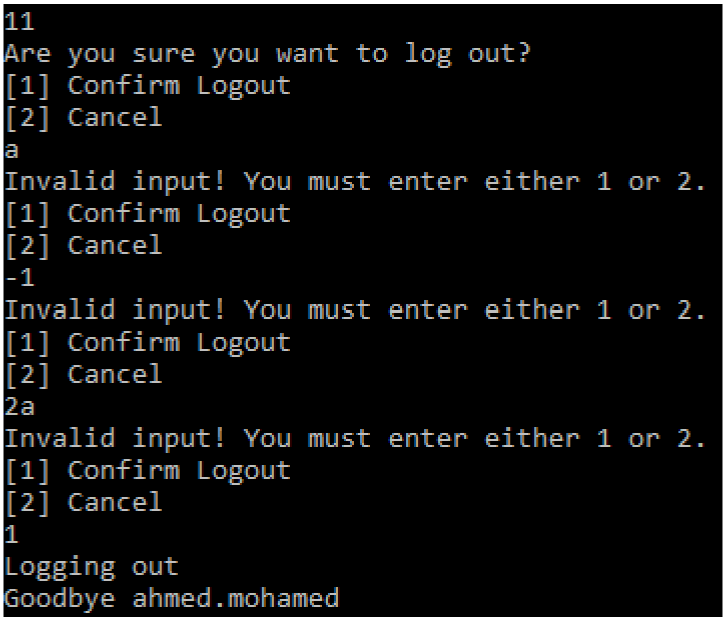
Input:

Must be either 1 or 2.

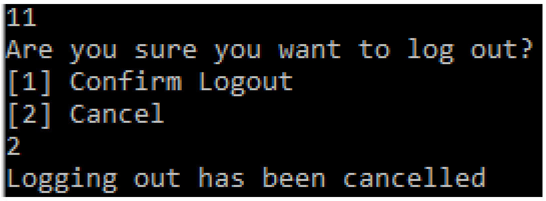
Mechanism:

- The function prints the 2 options: logging out or cancelling.
- The function uses the function `readInteger` to get input from the user.
- While the input is neither 1 nor 2, an error message is printed.
- If the input is 1, then the variable `logged_in` is set to false.
- If the input is 2, then the function returns to the `menu` function without toggling `logged_in`.

Test sample:



```
11
Are you sure you want to log out?
[1] Confirm Logout
[2] Cancel
a
Invalid input! You must enter either 1 or 2.
[1] Confirm Logout
[2] Cancel
-1
Invalid input! You must enter either 1 or 2.
[1] Confirm Logout
[2] Cancel
2a
Invalid input! You must enter either 1 or 2.
[1] Confirm Logout
[2] Cancel
1
Logging out
Goodbye ahmed.mohamed
```



```
11
Are you sure you want to log out?
[1] Confirm Logout
[2] Cancel
2
Logging out has been cancelled
```

6.3. Void quit()

Description:

The function `quit` is responsible for quitting the program.

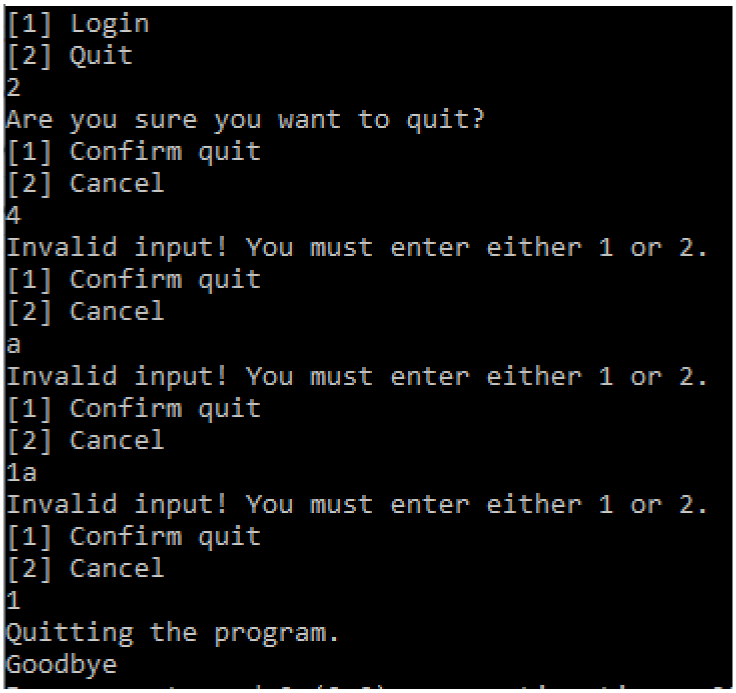
Input:

Must be either 1 or 2 only.

Mechanism:

- The function prints the 2 options: logging out or cancelling.
- The function uses the function `readInteger` to get input from the user.
- While the input is neither 1 nor 2, an error message is printed.
- If the input is 1, then the program is exited.
- If the input is 2, then the function returns back to the `menu` function.

Test sample:



```
[1] Login
[2] Quit
2
Are you sure you want to quit?
[1] Confirm quit
[2] Cancel
4
Invalid input! You must enter either 1 or 2.
[1] Confirm quit
[2] Cancel
a
Invalid input! You must enter either 1 or 2.
[1] Confirm quit
[2] Cancel
1a
Invalid input! You must enter either 1 or 2.
[1] Confirm quit
[2] Cancel
1
Quitting the program.
Goodbye
```

6.4. Void menu()

Description:

The function `menu` is the only function called in the `main` function and all other functions are called from within it. It shows actually two different menus depending on the variable `logged_in`.

In case `logged_in` is equal to false:

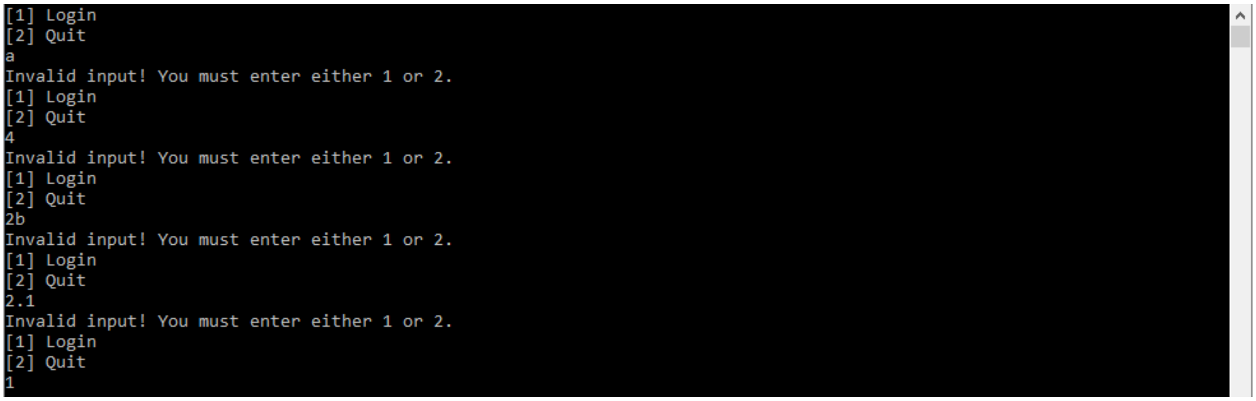
Input:

Must be either 1 or 2 only.

Mechanism:

- The function prints the 2 options: log in or quit.
- The function uses the function `readInteger` to get input from the user.
- While the input is neither 1 nor 2, an error message is printed.
- If the input is 1, then the function `login` is called.
- If the input is 2, then the function `quit` is called.

Test sample:



```
[1] Login
[2] Quit
a
Invalid input! You must enter either 1 or 2.
[1] Login
[2] Quit
4
Invalid input! You must enter either 1 or 2.
[1] Login
[2] Quit
2b
Invalid input! You must enter either 1 or 2.
[1] Login
[2] Quit
2.1
Invalid input! You must enter either 1 or 2.
[1] Login
[2] Quit
1
```

In case `logged_in` is equal to false:

Input:

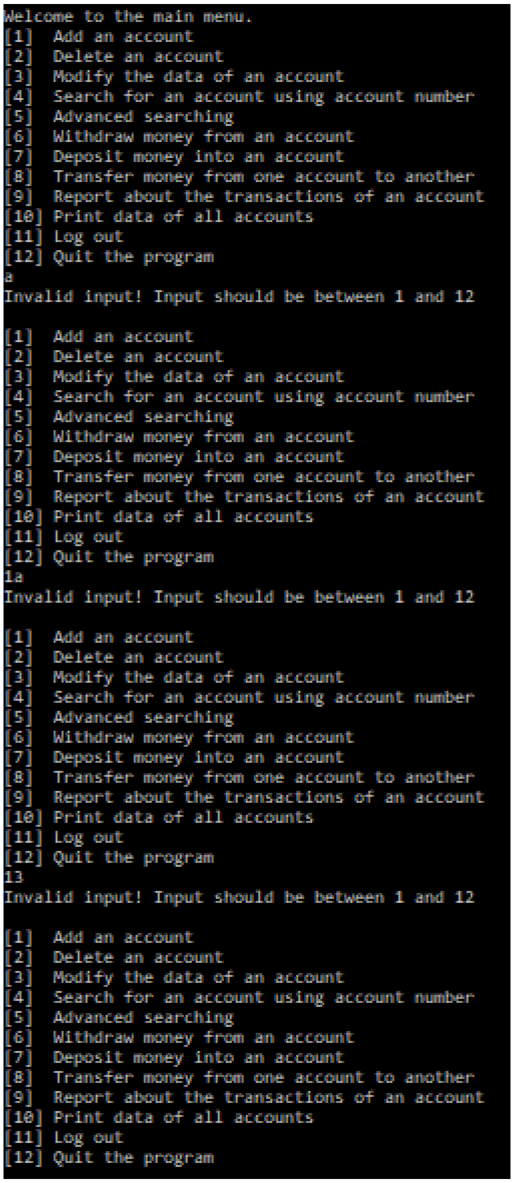
Must be an integer value between 1 and 12.

Mechanism:

- The function prints the 12 options.
- The function uses the function `readInteger` to get input from the user.
- While the input is not within the range of 1 to 12, an error message is printed.
- Each number corresponds to a function that is called when that number is entered.
- If the input is 1, then the function `add` is called.
- If the input is 2, then the function `delete_account` is called.
- If the input is 3, then the function `modify_acc` is called.
- If the input is 4, then the function `query_search` is called.
- If the input is 5, then the function `advanced_search` is called.
- If the input is 6, then the function `withdraw` is called.
- If the input is 7, then the function `deposit` is called.
- If the input is 8, then the function `transfer` is called.
- If the input is 9, then the function `report` is called.
- If the input is 10, then the function `print` is called.
- If the input is 11, then the function `log_out` is called.

- If the input is 12, then the function `quit` is called.

Test sample:



```
Welcome to the main menu.
[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
a
Invalid input! Input should be between 1 and 12

[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
1a
Invalid input! Input should be between 1 and 12

[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
13
Invalid input! Input should be between 1 and 12

[1]  Add an account
[2]  Delete an account
[3]  Modify the data of an account
[4]  Search for an account using account number
[5]  Advanced searching
[6]  Withdraw money from an account
[7]  Deposit money into an account
[8]  Transfer money from one account to another
[9]  Report about the transactions of an account
[10] Print data of all accounts
[11] Log out
[12] Quit the program
```

7. *USER MANUAL*

The user can return to main menu at anytime in the code by entering “-1”.

Exit Codes:

0	Exit with no error
1	Couldn’t find/open users.txt file
2	Couldn’t find/open accounts.txt file
3	Couldn’t find/open accountNo.txt file
4	To many invalid inputs in login

8. *REFERENCES*

GitHub Repository:

https://github.com/AhmWael/Bank_Management_System_Project