

## Test 3

You are tasked with creating PhoneBook in C++ that can be used to store and manage personal contacts. The program allows users to add, view, delete, save, and load contacts. Additionally, the system includes a basic security feature that requires users to enter a 4-digit prime number as a password to access the program. The program can store up to 100 contacts in memory and can save and load them from a binary file called "contacts.dat"

```
struct Contact
{
    string name;
    string phoneNumber;
};
```

### Instructions

- 1 Declare a struct named '**Contact**' with the following data members:
  - '**name**' – A string to the contacts name and surname.
  - '**phoneNumber**' – A string to hold the contact's phone number
- 2 Write code for the following functions:
  - 2.1 The program starts by prompting the user to enter a 4-digit prime number as a password. If the user enters an invalid password, the program should keep prompting until a valid password is provided. **Create a function to check if the password is prime**
  - 2.2 A function named '**addContact**' that allows the user to add a new contact by providing a name and phone number. It checks for the maximum contact limit and validates the phone number format.
    - i. A function named '**isValidPhoneNumber**' which checks whether a given phone number follows the format xxx-xxx-xxxx/072-000-0000 using regular expressions. It returns a boolean indicating the validity of the phone number.  
Hint: include the necessary header for regular expressions **regex**

```
regex pattern("[0]\\d{2}-\\d{3}-\\d{4}$");  
return regex_match(phoneNumber , pattern);
```
  - 2.3 A function named '**viewAllContacts**' which displays all stored contacts to the user, including their names and phone numbers.

2.4 A function named '**searchContactByName**' which enables users to search for contacts by name and optionally delete them. It displays search results and handles user input for deletion.

2.5 A function named '**saveContactsToFile**' which saves the contact data stored in an array to a binary file named "contacts.dat." It checks for errors in file opening and writes contact data to the file.

2.6 A function named '**loadContactsFromFile**' which loads contact data from the "contacts.dat" binary file into an array. It updates the count of contacts read from the file and handles situations where the file is not found or is empty.

### 3 In the '**main()**' function:

To complete the code the main function serves as the program's entry point. It handles user authentication, menu options, and the overall program flow. Users can perform actions like adding contacts, viewing contacts, searching & deleting contacts, saving to a file, loading from a file, and logging out. The program runs in a loop until the user chooses to exit, and prompts the user to enter the password again.

### Example program execution

```
Enter a 4-digit prime number as a password to access the program (1009, 2063, 3911, 6571, etc): 1004
Invalid password. Please enter a valid 4-digit prime number.
Enter a 4-digit prime number as a password to access the program (1009, 2063, 3911, 6571, etc): 2063
```

```
Contact Management System
1. Add Contact
2. View All Contacts
3. Delete Contact by Name
4. Save to SimCard
5. Load from SimCard
6. Logout
Enter your choice:
```

```
Contact Management System
1. Add Contact
2. View All Contacts
3. Delete Contact by Name
4. Save to SimCard
5. Load from SimCard
6. Logout
Enter your choice: 1

Enter contact name: Jan Potchefstroom
Enter contact phone number (xxx-xxx-xxxx): 018-444-4444
Contact added successfully.
```

```
Enter contact name: May Rustenburg
Enter contact phone number (xxx-xxx-xxxx): 874-999-9999
Invalid phone number format. Please enter a valid phone number (xxx-xxx-xxxx).
```

Contact Management System

1. Add Contact
2. View All Contacts
3. Delete Contact by Name
4. Save to SimCard
5. Load from SimCard
6. Logout

Enter your choice: 2

All Contacts:

Contact #1: Name: Jan Potchefstroom, Phone Number: 018-444-4444  
Contact #2: Name: Feb Vanderbijlpark, Phone Number: 016-222-2222  
Contact #3: Name: March Mahikeng, Phone Number: 018-777-7777  
Contact #4: Name: April Potchefstroom, Phone Number: 018-555-5555

Contact Management System

1. Add Contact
2. View All Contacts
3. Delete Contact by Name
4. Save to SimCard
5. Load from SimCard
6. Logout

Enter your choice: 3

Enter contact name to search: Potchefstroom

Search Results:

Contact #1: Name: Jan Potchefstroom, Phone Number: 018-444-4444  
Contact #2: Name: April Potchefstroom, Phone Number: 018-555-5555  
Do you want to delete any of these entries? (y/n): y  
Enter the index of the entry to delete (1-2): 1  
Contact deleted successfully.

All Contacts:

Contact #1: Name: Feb Vanderbijlpark, Phone Number: 016-222-2222  
Contact #2: Name: March Mahikeng, Phone Number: 018-777-7777  
Contact #3: Name: April Potchefstroom, Phone Number: 018-555-5555





Contact Management System

1. Add Contact
2. View All Contacts
3. Delete Contact by Name
4. Save to SimCard
5. Load from SimCard
6. Logout

Enter your choice: 4

Saving contacts to a binary file...

Contacts saved successfully.

 bin	2023/10/10 22:03	File folder
 obj	2023/10/10 22:03	File folder
 contacts.dat	2023/10/11 02:04	DAT File
 main.cpp	2023/10/11 01:51	C++ source file

```

Enter a 4-digit prime number as a password to access the program (1009, 2063, 3911, 6571, etc): 2063

Contact Management System
1. Add Contact
2. View All Contacts
3. Delete Contact by Name
4. Save to SimCard
5. Load from SimCard
6. Logout
Enter your choice: 5
Loading contacts from a binary file...
Contacts loaded successfully.

Contact Management System
1. Add Contact
2. View All Contacts
3. Delete Contact by Name
4. Save to SimCard
5. Load from SimCard
6. Logout
Enter your choice: 2

All Contacts:
Contact #1: Name: Feb Vanderbijlpark, Phone Number: 016-222-2222
Contact #2: Name: March Mahikeng, Phone Number: 018-777-7777
Contact #3: Name: April Potchefstroom, Phone Number: 018-555-5555

```

	Item	Mark allocation
Q1	<b>Successfully authenticated with a valid password (10 Marks)</b>	
	[5 Marks] - Authentication requires a 4-digit prime number.	5
	[5 Marks] - Effective handling of incorrect password inputs.	5
	<b>Successfully adds a contact to the list (8 Marks)</b>	
	[3 Marks] - Accepts contact name and phone number.	3
	[5 Marks] - Validates phone number format and displays appropriate messages.	5
	<b>Successfully lists all contacts (7 Marks)</b>	
	[4 Marks] - Lists contacts.	4
	[3 Marks] - Displays contact details correctly.	3
	<b>Successfully deletes contacts by name (8 Marks)</b>	
	[4 Marks] - Searches for contacts by name.	4
	[4 Marks] - Allows deletion of contacts with proper input validation.	4
	<b>Successfully saves contacts to a file (10 Marks)</b>	
	[5 Marks] - Writes contacts to a binary file.	5
	[5 Marks] - Handles file opening and writing errors gracefully.	5
	<b>Successfully loads contacts from a file (10 Marks)</b>	
	[5 Marks] - Reads contacts from a binary file.	5
	[5 Marks] - Handles file not found or empty file situations.	5
	<b>Code is well-organized and modular, making use of functions for various actions (5 Marks)</b>	
	[5 Marks] - Code is moderately organized with separate functions for major actions.	5
	<b>Code Comments (2 Marks)</b>	
	[2 Marks] - Adequate comments explaining code functionality.	2

	Comments included in program (penalty 2), file name according to the specification (penalty 2), correct file extension (project zipped) (penalty – not marked), Logical flow and readability of the code (penalty range from 1 to 5, depending on severity).	
	<b>Total</b>	<b>60</b>