8.	To illustrate the concept of inheritance and its types in C++. (WAP of your choice)	2081/04/25	
9.	To illustrate the concept of public, protected and private keyword in inheritance. (WAP of your choice)	2081/04/25	
10.	To illustrate the concept of pure virtual functions and abstract classes in C++. (WAP of your choice)	2081/04/25	
11.	To illustrate the concept of runtime polymorphism in C++. (WAP of your choice)	2081/04/25	
12.	Lab 12: To study the stream classes and File Handling in C++. a. Write a program to demonstrate the stream operators (insertion and extraction) overloading. b. Write a C++ program to enter the names of any five person and store in a text file named "person.txt". c. Write a C++ program to display the content of "person.txt" file on the console. d. Write a C++ program to add more new records in the file "person.txt". e. Write a C++ program to copy the contents of one text file to another. The program should read from "source.txt" and write the content to "destination.txt". f. Write a C++ program that searches for a specific word in a text file and counts the number of times it appears. The program should be case-insensitive. g. Write a C++ program to encrypt the content of a file using a simple algorithm (e.g., Caesar cipher) and write the encrypted content to another file. Then, write a function to decrypt the file and display the original content.	2081/04/25	
13.	Development of a Simple Console Based App using C++.	2081/04/25	

LAB 8

OBJECTIVE: To illustrate the concept of inheritance and its types in C++.

THEORY:

Inheritance in C++ is a feature that allows a class (called the derived class) to inherit properties and behaviors (i.e., members and functions) from another class (called the base class). This allows for code reuse and establishing relationships between classes.

Types of Inheritance in C++:

- **Single Inheritance:** A class inherits from one base class.
- **Multiple Inheritance:** A class inherits from more than one base class.
- **Multilevel Inheritance:** A class is derived from another derived class.
- **Hierarchical Inheritance:** Multiple classes are derived from a single base class.

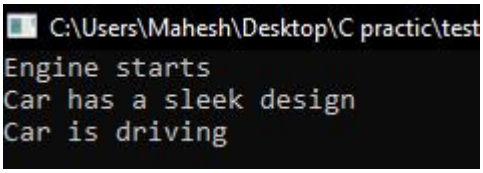
PROGRAMS:

a. Program to illustrate Single inheritance in C++.

```
#include <iostream>
using namespace std;
class Animal {
public:     void sound() {
                cout << "Animal makes sound" << endl;
                }
};
class Dog : public Animal {
public:     void bark() {
                cout << "Dog barks" << endl;
                }
};
int main() {
    Dog myDog;
    myDog.sound(); // Inherited from Animal
    myDog.bark();    // Specific to Dog
    return 0;
}
```

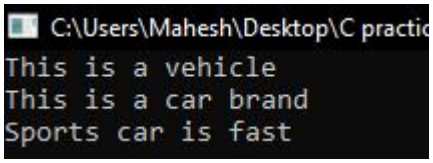
b. Program to illustrate Multiple inheritance in C++.

```
#include <iostream>
using namespace std;
class Animal {
public:         void sound() {
                        cout << "Animal makes sound" << endl;
                        }
};
class Dog : public Animal {
public:         void bark() {
                        cout << "Dog barks" << endl;
                }
};
int main() {
    Dog myDog;
    myDog.sound(); // Inherited from Animal
    myDog.bark();    // Specific to Dog
    return 0;
}
```



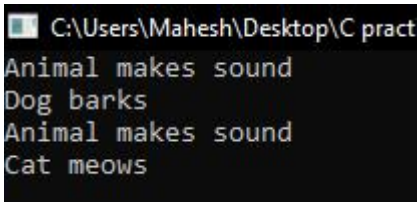
c. Program to illustrate Multilevel inheritance in C++.

```
#include <iostream>
using namespace std;
class Vehicle {
public:		void type() {
                        cout << "This is a vehicle" << endl;
                }
};class Car : public Vehicle {
public:		void brand() {
                        cout << "This is a car brand" << endl;
                }
};class SportsCar : public Car {
public:		void speed() {
                        cout << "Sports car is fast" << endl;
                }
};
int main() {
        SportsCar myCar;
        myCar.type();    // Inherited from Vehicle
        myCar.brand(); // Inherited from Car
        myCar.speed(); // Specific to SportsCar
        return 0;
}
```



d. Program to illustrate Hierarchical inheritance in C++.

```
#include <iostream>
using namespace std;
class Animal {
public:	void sound(){
                        cout << "Animal makes sound" << endl;
                }
};class Dog : public Animal {
public:	void bark() {
                        cout << "Dog barks" << endl;
                }
};class Cat : public Animal {
public:		void meow() {
                        cout << "Cat meows" << endl;
                }
};
int main() {
        Dog myDog;
        Cat myCat;

        myDog.sound(); // Inherited from Animal
        myDog.bark();    // Specific to Dog

        myCat.sound(); // Inherited from Animal
        myCat.meow();    // Specific to Cat
        return 0;
}
```



CONCLUSION:

This laboratory exercise provided a hands-on experience in C++ program. Students gained practical knowledge of implementing basic of inheritance in C++ programming and are now better equipped to undertake more complex programming tasks in the future. By understanding the syntax and use of inheritance, we can effectively leverage them in our C++ programs when necessary.

LAB 9

OBJECTIVE:

To illustrate the concept of public, protected and private keyword in inheritance.

THEORY:

In C++, the access specifiers public, protected, and private define how members (attributes and methods) of a class can be accessed in relation to other classes and objects.

- Public: Members declared as public are accessible from outside the class and in derived classes.
- Protected: Members declared as protected are not accessible outside the class but are accessible in derived classes.
- Private: Members declared as private are only accessible within the same class and are not accessible in derived classes.

When using inheritance in C++, the access specifiers for the base class can affect how its members are accessed in the derived class.

PROGRAMS:

```
#include <iostream>
using namespace std;

class Base {
public:         int publicVar;
protected:      int protectedVar;
private:        int privateVar;

public:
    Base() {
        publicVar = 10;
        protectedVar = 20;
        privateVar = 30;
    }
    void displayBase() {
        cout << "Base Class - Public: " << publicVar << ", Protected: " << protectedVar << ",
Private: " << privateVar << endl;
    }
};

// Derived class using public inheritance
class Derived : public Base {
public:
    void displayDerived() {  // Accessing members from Base class
        cout << "Derived Class - Public: " << publicVar << endl;             // Accessible
        cout << "Derived Class - Protected: " << protectedVar << endl; // Accessible
        // cout << "Private: " << privateVar; // Not Accessible
    }
};

int main() {
    Base baseObj;
    Derived derivedObj;

    // Accessing members of Base from outside
    cout << "Outside Base Class - Public: " << baseObj.publicVar << endl; // Accessible
    // cout << baseObj.protectedVar; // Not Accessible
    // cout << baseObj.privateVar;     // Not Accessible
```

```
    baseObj.displayBase();          // Accessing all members within the Base class
    derivedObj.displayDerived(); // Accessing public and protected members in Derived class

    return 0;
}
```

```
C:\Users\Mahesh\Desktop\C practic\test.exe
Outside Base Class - Public: 10
Base Class - Public: 10, Protected: 20, Private: 30
Derived Class - Public: 10
Derived Class - Protected: 20
```

EXPLANATION:

1. Class Base:
   - Contains public, protected, and private members.
   - The constructor initializes these members.
   - The displayBase() function prints all members, since a class can access all its own members, including private ones.

2. Class Derived:
   - Inherits from the Base class using public inheritance.
   - Inside displayDerived(), it can access the public and protected members from the base class but cannot access the private members.

3. main() function:
   - Demonstrates how the public, protected, and private members are accessed:
   - publicVar is accessible outside the class and in derived classes.
   - protectedVar is accessible in derived classes but not outside the class.
   - privateVar is not accessible outside the class or in derived classes.

CONCLUSION:

The experiment demonstrates the use of public, protected, and private access specifiers in C++ inheritance. It shows that:

- Public members are accessible everywhere.
- Protected members are accessible in derived classes but not outside the class.
- Private members are only accessible within the class that declares them, not in derived classes or from outside.

This illustrates the control over data accessibility in different inheritance scenarios.

LAB 10

OBJECTIVE:

To illustrate the concept of pure virtual functions and abstract classes in C++.

THEORY:

- **Abstract Class:** A class that contains at least one pure virtual function is called an abstract class. It cannot be instantiated directly.
- **Pure Virtual Function:** A virtual function is made pure by assigning = 0 in its declaration. Any class containing a pure virtual function must be inherited and its pure virtual function must be overridden.

PROGRAMS:

Here the program, in which parent class Person is created with pure virtual functions for setting and displaying properties like getName() and getAge(). A child class Child inherits from the Person class and overrides these functions to provide specific details for a child.

```
#include <iostream>
using namespace std;

class Person {
public:
    virtual string getName() = 0;    virtual int getAge() = 0;
};
class Child : public Person {
private:
    string name;
    int age;
public:
    Child(string name, int age) : name(name), age(age) {}
    string getName() {    // Override pure virtual functions
        return name;
    }
    int getAge() {
        return age;
    }
    void displayInfo() {
        cout << "Child's Name: " << getName() << endl;
        cout << "Child's Age: " << getAge() << endl;
    }
};
int main() {
    Child child1("Rohan", 10);
    child1.displayInfo();
    return 0;
}
```

```
C:\Users\Mahesh\Desktop\C practic\test.exe
Child's Name: Rohan
Child's Age: 10
```

CONCLUSION:

In this program, the Person class acts as the abstract parent class with pure virtual functions getName() and getAge(). The Child class overrides these functions to provide specific details for a child. This demonstrates how a parent class can define abstract properties, which must be implemented in derived child classes.

This program clearly shows how to create an abstract parent class with properties and then override them in the child class to achieve specific functionality.

LAB 11

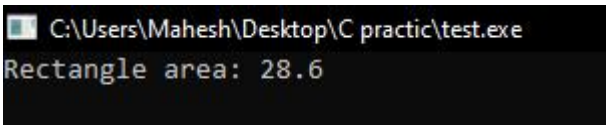
OBJECTIVE: To illustrate the concept of runtime polymorphism in C++.

THEORY:

- **Polymorphism**: It allows objects to be treated as instances of their parent class, even if they are actually instances of derived classes.
- **Runtime Polymorphism**: Achieved using virtual functions and pointers/references, where the function call is resolved at runtime, allowing different derived class implementations to be invoked.
- **Virtual Function**: A function declared in the base class using the keyword virtual and overridden in derived classes.

PROGRAMS:

```
#include <iostream>
using namespace std;

class Shape {   // Base class with a virtual function
public:
    virtual void area() {
        cout << "Calculating area of shape..." << endl;
    }
};
class Rectangle : public Shape {
    double length, width;
public:
    Rectangle(double l, double w) : length(l), width(w) {}

    void area() {
        cout << "Rectangle area: " << length * width << endl;
    }
};
int main() {
    Shape* shape;    // Base class pointer
    Rectangle rectangle(4.4, 6.5);    // Derived class object Rectangle

    // Runtime polymorphism using base class pointer
    shape = &rectangle;
    shape->area();    // Calls Rectangle's area

    return 0;
}
```



CONCLUSION:

The program demonstrates runtime polymorphism using virtual functions and base class pointers. When the base class pointer points to different derived class objects, the appropriate overridden method is invoked, which is determined during program execution (runtime). This allows for dynamic behavior and flexibility in handling objects of different derived types through a common interface.

This lab report illustrates the core concept of runtime polymorphism and how it can be implemented in C++ using virtual functions.

LAB 12

OBJECTIVE: To study the stream classes and File Handling in C++.

THEORY:

C++ provides classes to perform input and output operations using files. These classes are part of the <fstream> library, and the commonly used classes include:

- ifstream: For reading from files (input file stream).
- ofstream: For writing to files (output file stream).
- fstream: For both reading and writing to files.

These classes are derived from the base class iostream. The file handling in C++ involves:

1. Opening the file using open() or using the constructor.
2. Performing read/write operations.
3. Closing the file using close().

Types of File Modes:

- ios::in: Open file for reading.
- ios::out: Open file for writing.
- ios::app: Open file in append mode.
- ios::binary: Open file in binary mode.

PROGRAMS:

a. Write a program to demonstrate the stream operators (insertion and extraction) overloading.

```
#include <iostream>
using namespace std;

class Person {
public:
    string name;
    int age;
    friend ostream& operator<<(ostream& out, const Person& p);
    friend istream& operator>>(istream& in, Person& p);
};
ostream& operator<<(ostream& out, const Person& p) {
    out << "Name: " << p.name << ", Age: " << p.age;
    return out;
}
istream& operator>>(istream& in, Person& p) {
    cout << "Enter name and age: ";
    in >> p.name >> p.age;
    return in;
}
int main() {
    Person p;
    cin >> p;
    cout << p;
    return 0;
}
```

```
C:\Users\Mahesh\Desktop\C practic\test.exe
Enter name and age: Mahesh
21
Name: Mahesh, Age: 21
--------------------------------
```

b. Write a C++ program to enter the names of any five person and store in a text file named "person.txt".

```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    ofstream file("person.txt");
    for (int i = 0; i < 5; i++) {
        string name;
        cout << "Enter name: ";
        cin >> name;
        file << name << endl;
    }
    file.close();
    return 0;
}
```

```
C:\Users\Mahesh\Desktop\C practic\test.exe
Enter name: Mahesh
Enter name: Ramesh
Enter name: Suresh
Enter name: Ganesh
Enter name: Manish
```

```
person.txt - Not
File  Edit  Forma
Mahesh
Ramesh
Suresh
Ganesh
Manish
```

c. Write a C++ program to display the content of "person.txt" file on the console.

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main() {
    ifstream file("person.txt");
    string line;
    while (getline(file, line)) {
        cout << line << endl;
    }
    file.close();
    return 0;
}
```

```
C:\Users\
Mahesh
Ramesh
Suresh
Ganesh
Manish
```

d. Write a C++ program to add more new records in the file "person.txt".

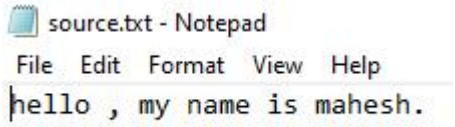
```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    ofstream file("person.txt", ios::app);
    string name;
    cout << "Enter new name: ";
    cin >> name;
    file << name << endl;
    file.close();
    return 0;
}
```

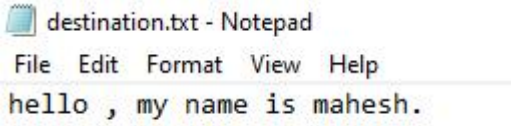
```
C:\Users\Mahesh\Desktop\C practic\te
Enter new name: Nitesh
```

```
person.txt
File  Edit  F
Mahesh
Ramesh
Suresh
Ganesh
Manish
Nitesh
```

e. Write a C++ program to copy the contents of one text file to another. The program should read from “source.txt” and write the content to “destination.txt”.

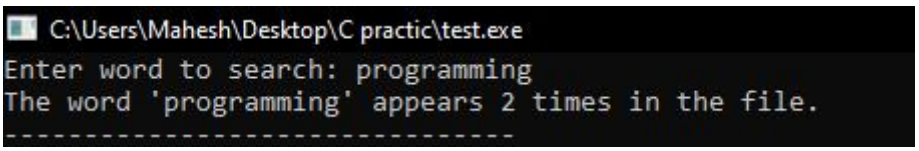
```
#include <iostream>
#include <fstream>
using namespace std;

int main() {
    ifstream src("source.txt");
    ofstream dest("destination.txt");
    string line;
    while (getline(src, line)) {
        dest << line << endl;
    }
    src.close();
    dest.close();
    return 0;
}
```





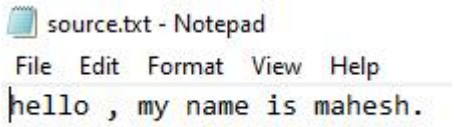
f. Write a C++ program that searches for a specific word in a text file and counts the number of times it appears. The program should be case-insensitive.

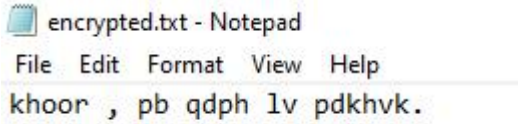
```
#include <iostream>
#include <fstream>
#include <string>
#include <algorithm>
using namespace std;

int main() {
    ifstream file("file.txt");
    string word, temp;
    int count = 0;
    cout << "Enter word to search: ";
    cin >> word;
    transform(word.begin(), word.end(), word.begin(), ::tolower);

    while (file >> temp) {
        transform(temp.begin(), temp.end(), temp.begin(), ::tolower);
        if (temp == word) {
            count++;
        }
    }
    cout << "The word '" << word << "' appears " << count << " times in the file.";
    file.close();
    return 0;
}
```

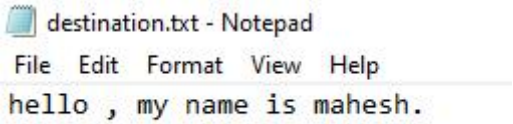


g. Write a C++ program to encrypt the content of a file using a simple algorithm (e.g., Caesar cipher) and write the encrypted content to another file. Then, write a function to decrypt the file and display the original content.

```
#include <iostream>
#include <fstream>
using namespace std;

void encrypt(char sourceFile[], char encryptedFile[], int shift) {
	ifstream src(sourceFile);
	ofstream enc(encryptedFile);
	char c;
	while (src.get(c)) {
		if (isalpha(c)) {
			char base = islower(c) ? 'a' : 'A';
			c = (c - base + shift) % 26 + base;
		}
		enc << c;
	}
	src.close();
	enc.close();
}

void decrypt (char encryptedFile[], char decryptedFile[], int shift) {
	ifstream enc(encryptedFile);
	ofstream dec(decryptedFile);
	char c;
	while (enc.get(c)) {
		if (isalpha(c)) {
			char base = islower(c) ? 'a' : 'A';
			c = (c - base - shift + 26) % 26 + base;
		}
		dec << c;
	}
	enc.close();
	dec.close();
}

int main() {
	encrypt("source.txt", "encrypted.txt", 3);
	decrypt("encrypted.txt", "decrypted.txt", 3);
	return 0;
}
```







Conclusion:

The tasks explored in this lab demonstrate the capabilities of C++ stream classes and file handling, covering basic file operations, content management, and encryption techniques. These examples provide a practical understanding of how to manipulate file data effectively using C++.

LAB 13

OBJECTIVE: Development of a Simple Console Based App using C++.

BANK MANAGEMENT SYSTEM

Introduction:

The Bank Management System is designed to simulate basic banking operations such as account creation, deletion, deposits, withdrawals, and account searches. This system interacts with the user via a menu-driven interface, allowing for efficient management of account data.

Features and Functionalities:

1. **Create New Account:**
   - The user can create a new bank account by providing personal information such as name, age, gender, and address.
   - The account is assigned a unique account number automatically.
   - The user must specify the account type (current or savings) and the opening balance (which cannot be negative).
   - The information is stored in the accounts.txt file.

2. **Delete Account:**
   - The user can delete an existing account by entering the account number.
   - The program retrieves and displays account details for confirmation.
   - After confirmation, the account is removed from the file.

3. **Withdraw Amount:**
   - Users can withdraw money from their account by providing the account number and the withdrawal amount.
   - The balance is updated, and the transaction is denied if the withdrawal amount exceeds the current balance.

4. **Add Amount:**
   - Users can deposit money into their account by providing the account number and the deposit amount.
   - The balance is updated, and the deposit is added to the account.

5. **Display All Records:**
   - This option allows the user to view a list of all accounts, including account number, name, age, gender, address, account type, and current balance.

6. **Search Account:**
   - Users can search for a specific account either by account number or by name.
   - If found, the account details are displayed.

7. **Exit:**
   - The program ends with a friendly exit message.

Modules

- **BankAccount Class:**
  i. Contains attributes like accountNumber, name, age, gender, address, accountType, and balance.
  ii. Provides methods for setting and retrieving these attributes, and for comparing account names during searches.

- **File Handling:**
  i. All account data is stored in a text file (accounts.txt).
  ii. The program uses file I/O operations to read and write account information, ensuring that data persists between program executions.

- **Dashboard Function:**
  i. Displays the main menu and allows the user to choose an operation to perform.
  ii. Each operation is handled in a separate function for clarity and modularity.

- **Header and Formatting:**
  i. Neat formatting and headers are provided for the different sections to enhance readability.
  ii. Data is displayed in a tabular format with appropriate spacing and alignment.

Screenshorts:

```
##########################################################################################
#                                                                                        #
#                        Welcome to Bank Management System                               #
#                                                                                        #
##########################################################################################

             Dashboard...

             1. Create New Account

             2. Delete Account

             3. Withdraw Amount

             4. Add Amount

             5. Display Records

             6. Search Account Record

             7. Exit


             Enter your choice:
```

```
##########################################################################################
#                                                                                        #
#                        Welcome to Bank Management System                               #
#                                                                                        #
##########################################################################################

              New Account Create Section...

     New account number is: 121

     Account type [c/s]:s

     Enter Basic details of new Account Holder
     Name: Nitesh Kumar
     Age: 20
     Gender[m/f/o]:m
     Address: Nepal

     Enter Opening Balance: Rs.500

     New Account created successfully...
```

```
  ##########################################################################################
  #                                                                                        #
  #                        Welcome to Bank Management System                               #
  #                                                                                        #
  ##########################################################################################

                Account Delete Section...

                Enter Acc. Number: 120
+---------+-----------------------+-----+---------+--------------+---------+--------------+
| AccNo.  | AccHolder Name        | Age | Gender  | Address      | AccType | Balance      |
+---------+-----------------------+-----+---------+--------------+---------+--------------+
| 120     | Mahesh Udas           | 21  | Male    | Birgunj      | Saving  | 125          |
+---------+-----------------------+-----+---------+--------------+---------+--------------+

                Do you want to Delete this Account [y/n]: s

                !!!! Invalid Choice  !!!!

                Do you want to Delete this Account [y/n]: y

                Account Deleted successfully....
```

```
  ##########################################################################################
  #                                                                                        #
  #                        Welcome to Bank Management System                               #
  #                                                                                        #
  ##########################################################################################

                Amount Widthdraw Section...

                Enter Acc. Number: 100
+---------+-----------------------+-----+---------+--------------+---------+--------------+
| AccNo.  | AccHolder Name        | Age | Gender  | Address      | AccType | Balance      |
+---------+-----------------------+-----+---------+--------------+---------+--------------+
| 100     | Mahesh Udas           | 21  | Male    | Pokhara      | Saving  | 51000        |
+---------+-----------------------+-----+---------+--------------+---------+--------------+

                Enter Amount to widthdraw: 5000

                Balance Widthdraw Successful...
                New Balance is: 46000
```

```
  ##########################################################################################
  #                                                                                        #
  #                        Welcome to Bank Management System                               #
  #                                                                                        #
  ##########################################################################################

                Amount Add Section...

                Enter Acc. Number: 100
+---------+-----------------------+-----+---------+--------------+---------+--------------+
| AccNo.  | AccHolder Name        | Age | Gender  | Address      | AccType | Balance      |
+---------+-----------------------+-----+---------+--------------+---------+--------------+
| 100     | Mahesh Udas           | 21  | Male    | Pokhara      | Saving  | 46000        |
+---------+-----------------------+-----+---------+--------------+---------+--------------+

                Enter Amount to Add: 500

                Balance Added Successful...
                New Balance is: 46500
```

```
##########################################################################################
#                                                                                        #
#                        Welcome to Bank Management System                               #
#                                                                                        #
##########################################################################################

                Records Section...

+---------+-----------------------+-----+---------+--------------+---------+--------------+
| AccNo.  | AccHolder Name        | Age | Gender  | Address      | AccType | Balance      |
+---------+-----------------------+-----+---------+--------------+---------+--------------+
| 100     | Mahesh Udas           | 21  | Male    | Pokhara      | Saving  | 46500        |
| 101     | Prashant Timalshina   | 20  | Male    | Pokhara-2    | Saving  | 5000         |
| 102     | Apeksha Thapa Magar   | 19  | Female  | pokhara      | Saving  | 9800         |
| 103     | Ashlesha Ghimira      | 25  | Female  | Pokhara      | Saving  | 500          |
| 104     | Bhuwan Awasthi        | 20  | Male    | Pokhara      | Saving  | 700          |
| 105     | Prabin Timalshina     | 19  | Male    | Pokhara      | Saving  | 500          |
| 107     | Aashish Saut          | 19  | Male    | Pokhara      | Current | 600          |
| 108     | Aaravi Dhakal         | 19  | Female  | Pokhara      | Saving  | 500          |
| 109     | Abhunav Lamichhane    | 20  | Male    | Pokhara      | Saving  | 800          |
| 111     | Ashish Chaudhary      | 25  | Male    | pokhara      | Saving  | 500          |
| 112     | Biraj Malla           | 19  | Male    | Nepal        | Saving  | 500          |
| 113     | Saurav Paudel         | 19  | Male    | Pokhara      | Current | 500          |
| 114     | Anil Paudel           | 20  | Male    | Pokhara      | Current | 600          |
| 115     | Samir Subedi          | 20  | Male    | Pokhara      | Saving  | 700          |
| 116     | Merina Jeral          | 19  | Female  | Nepal        | Saving  | 800          |
| 117     | Anand Thakur          | 21  | Male    | Janakpur     | Saving  | 600          |
| 118     | Nabin                 | 19  | Male    | Pokhara      | Current | 120          |
| 119     | Suraksha Adhikari     | 19  | Female  | Pokhara      | Saving  | 2            |
| 121     | Nitesh Kumar          | 20  | Male    | Nepal        | Saving  | 500          |
| 122     | Suresh Pathak         | 21  | Male    | pokhara      | Saving  | 100          |
+---------+-----------------------+-----+---------+--------------+---------+--------------+
```

```
##########################################################################################
#                                                                                        #
#                        Welcome to Bank Management System                               #
#                                                                                        #
##########################################################################################

                Search Account Section...

                1. AccNo.
                2. Name

                choose: 1

                Enter Account Number: 115
+---------+-----------------------+-----+---------+--------------+---------+--------------+
| AccNo.  | AccHolder Name        | Age | Gender  | Address      | AccType | Balance      |
+---------+-----------------------+-----+---------+--------------+---------+--------------+
| 115     | Samir Subedi          | 20  | Male    | Pokhara      | Saving  | 700          |
+---------+-----------------------+-----+---------+--------------+---------+--------------+
```

```
##########################################################################################
#                                                                                        #
#                        Welcome to Bank Management System                               #
#                                                                                        #
##########################################################################################

                Search Account Section...

                1. AccNo.
                2. Name

                choose: 2

                Enter Name to Search: Nitesh Kumar
+---------+-----------------------+-----+---------+--------------+---------+--------------+
| AccNo.  | AccHolder Name        | Age | Gender  | Address      | AccType | Balance      |
+---------+-----------------------+-----+---------+--------------+---------+--------------+
| 121     | Nitesh Kumar          | 20  | Male    | Nepal        | Saving  | 500          |
+---------+-----------------------+-----+---------+--------------+---------+--------------+
```

```
##########################################################################################
#                                                                                        #
#                        Welcome to Bank Management System                               #
#                                                                                        #
##########################################################################################

                      !!!!  Thankyou For your Time.  !!!!!
```

Conclusion:

This Bank Management System provides an effective way to simulate basic banking operations. It emphasizes ease of use, data persistence, and simple account management through file handling. The modular structure of the program allows for easy future modifications and additions, making it scalable for more advanced features in the future.