

```

/**
 * * Eshan Nahar
 * Implement a class Complex which represents the Complex Number data
type. Implement the following
 * 1. Constructor (Including a default constructor which creates
complex number 0+0i)
 * 2. Overload operator + to add two complex numebrs
 * 3. Overload operator * to multiply two complex numbers
 * 4. Overload operators << and >> to print and read complex numbers
 */
#include<bits/stdc++.h>
using namespace std;

class Complex {
private:
    double real;
    double imag;
public:
    Complex() : real(0), imag(0) {} // Default
    Complex(double r, double i) : real(r), imag(i) {} //
Parameterized
    // Other can be named anyhting
    // We don't have two parameters as it is a member function (this
object applies)
    Complex operator+(const Complex& other) { return Complex(real +
other.real, imag + other.imag); }
    Complex operator*(const Complex& other) {
        return Complex(real * other.real - imag * other.imag,
                        real * other.imag + imag * other.real);
    }

```

```

}

friend ostream& operator<<(ostream& os, const Complex& c) {
    os << c.real;
    if(c.imag >= 0) os << " + " << c.imag << "i";
    else os << " - " << -c.imag << "i";
    return os;
}

friend istream& operator>>(istream& is, Complex& c) {
    char plusMinus;
    is >> c.real >> plusMinus >> c.imag;
    if(plusMinus == '-') c.imag = -c.imag;
    return is;
}
};

```

```

int main() {
    cout << "Eshan Nahar" << endl;
    Complex a(0, 0);
    Complex b(0, 0);
    int choice;
    cout << "\nMenu:\n";
    cout << "1. Input Complex Number A\n";
    cout << "2. Input Complex Number B\n";
    cout << "3. Add A and B\n";
    cout << "4. Multiply A and B\n";
    cout << "5. Display A\n";
    cout << "6. Display B\n";
}

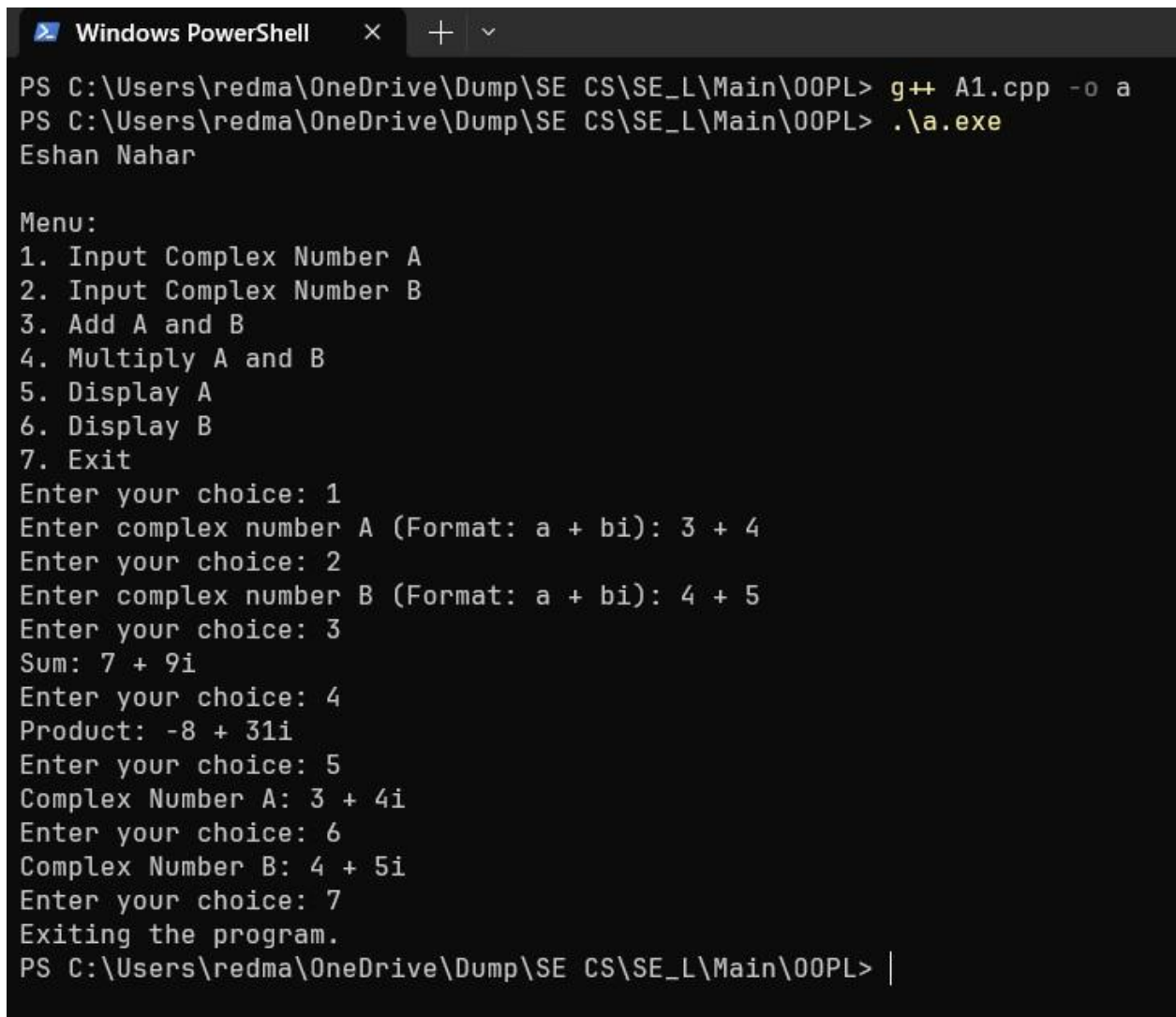
```

```

cout << "7. Exit\n";
do {
    cout << "Enter your choice: ";
    cin >> choice;
    switch (choice) {
        case 1: cout << "Enter complex number A (Format: a + bi): ";
cin >> a; break;
        case 2: cout << "Enter complex number B (Format: a + bi): ";
cin >> b; break;
        case 3: {
            Complex sum = a + b;
            cout << "Sum: " << sum << endl;
            break;
        }
        case 4: {
            Complex product = a * b;
            cout << "Product: " << product << endl;
            break;
        }
        case 5: cout << "Complex Number A: " << a << endl; break;
        case 6: cout << "Complex Number B: " << b << endl; break;
        case 7: cout << "Exiting the program." << endl; break;
        default: cout << "Invalid choice! Please try again." << endl;
    }
}
if (cin.fail()) {
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    cout << "Invalid input. Please try again." << endl;
}

```

```
    }  
    } while (choice != 7);  
    return 0;  
}
```



```
Windows PowerShell  x  +  v  
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> g++ A1.cpp -o a  
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> .\a.exe  
Eshan Nahar  
  
Menu:  
1. Input Complex Number A  
2. Input Complex Number B  
3. Add A and B  
4. Multiply A and B  
5. Display A  
6. Display B  
7. Exit  
Enter your choice: 1  
Enter complex number A (Format: a + bi): 3 + 4  
Enter your choice: 2  
Enter complex number B (Format: a + bi): 4 + 5  
Enter your choice: 3  
Sum: 7 + 9i  
Enter your choice: 4  
Product: -8 + 31i  
Enter your choice: 5  
Complex Number A: 3 + 4i  
Enter your choice: 6  
Complex Number B: 4 + 5i  
Enter your choice: 7  
Exiting the program.  
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> |
```

```
/**
```

```
 * * Eshan Nahar
```

```
 * Develop a program in C++ to create a database of student's
information system containing the following information: Name, Roll
number, Class, Division, Date of Birth, Blood group, Contact address,
Telephone number, Driving license no. and other. Construct the
database with suitable member function
```

```
 * Make use of constructor, default constructor, copy constructor,
destructor, static member functions, friend class, this pointer,
inline code and dynamic memory allocation operators-new and delete as
well as exception handling.
```

```
 */
```

```
#include <bits/stdc++.h>
```

```
using namespace std;
```

```
class Address {
```

```
    private:
```

```
        string street;
```

```
        string city;
```

```
        string state;
```

```
        string country;
```

```
        int zipCode;
```

```
    public:
```

```
        Address(string st, string c, string s, string co, int zip) :
street(st), city(c), state(s), country(co), zipCode(zip) {}
```

```
        friend ostream& operator<<(ostream& os, const Address& addr);
```

```
};
```

```
ostream& operator<<(ostream& os, const Address& addr) {
```

```

    os << addr.street << ", " << addr.city << ", " << addr.state << ",
" << addr.country << ", " << addr.zipCode;

    return os;
}

```

```

class Student {
    private:
        string name;
        int rollNumber;
        string className;
        char division;
        string dateOfBirth;
        string bloodGroup;
        Address* contactAddress;
        long telephoneNumber;
        string drivingLicenseNumber;
        static int totalStudents;

    public:
        Student(string n, int rn, string c, char d, string dob, string
bg, string st, string city, string state, string country, int zip,
long tel, string dl)
            : name(n), rollNumber(rn), className(c), division(d),
dateOfBirth(dob), bloodGroup(bg), telephoneNumber(tel),
drivingLicenseNumber(dl) {
            contactAddress = new Address(st, city, state, country, zip);
            totalStudents++;
        }
}

```

```

    Student() : name("Unknown"), rollNumber(0), className("Unknown"),
division('A'), dateOfBirth("01/01/1900"), bloodGroup("Unknown"),
telephoneNumber(0), drivingLicenseNumber("Unknown") {

    contactAddress = new Address("Unknown", "Unknown", "Unknown",
"Unknown", 0);

    totalStudents++;

}

```

```

    Student(const Student& other) : name(other.name),
rollNumber(other.rollNumber), className(other.className),
division(other.division), dateOfBirth(other.dateOfBirth),
bloodGroup(other.bloodGroup), telephoneNumber(other.telephoneNumber),
drivingLicenseNumber(other.drivingLicenseNumber) {

    contactAddress = new Address(*other.contactAddress);

    totalStudents++;

}

```

```

~Student() {

    delete contactAddress;

    totalStudents--;

}

```

```

static int getTotalStudents() { return totalStudents; }

friend ostream& operator<<(ostream& os, const Student& student);

inline int getRollNumber() const { return rollNumber; }

// Setter functions to update student information

void setName(const string& n) { name = n; }

void setRollNumber(int rn) { rollNumber = rn; }

void setClassName(const string& c) { className = c; }

void setDivision(char d) { division = d; }

```

```

        void setDateOfBirth(const string& dob) { dateOfBirth = dob; }
        void setBloodGroup(const string& bg) { bloodGroup = bg; }
        void setTelephoneNumber(long tel) { telephoneNumber = tel; }
        void setDrivingLicenseNumber(const string& dl) {
drivingLicenseNumber = dl; }

        void setContactAddress(const string& st, const string& city,
const string& state, const string& country, int zip) {
            delete contactAddress; // Free existing memory
            contactAddress = new Address(st, city, state, country, zip);
// Allocate new memory
        }
};

```

```

int Student::totalStudents = 0; // Resolution operator
ostream& operator<<(ostream& os, const Student& student) {
    os << "Name: " << student.name << endl;
    os << "Roll Number: " << student.rollNumber << endl;
    os << "Class: " << student.className << endl;
    os << "Division: " << student.division << endl;
    os << "Date of Birth: " << student.dateOfBirth << endl;
    os << "Blood Group: " << student.bloodGroup << endl;
    os << "Contact Address: " << *student.contactAddress << endl;
    os << "Telephone Number: " << student.telephoneNumber << endl;
    os << "Driving License Number: " << student.drivingLicenseNumber <<
endl;
    return os;
}

```

```

int main() {

```



```

    Student* students[100]; // Array to hold pointers to Student
objects
    int studentCount = 0;
    int choice;
    cout << "\nMenu:\n";
    cout << "1. Add Student\n";
    cout << "2. Update Student\n";
    cout << "3. Delete Student\n";
    cout << "4. Display All Students\n";
    cout << "5. Display Total Students\n";
    cout << "6. Exit\n";
    do {

        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1: {
                string name, className, dateOfBirth, bloodGroup, street,
city, state, country, drivingLicenseNumber;
                int rollNumber, zipCode;
                char division;
                long telephoneNumber;

                cout << "Enter Name (String): ";
                cin.ignore(); // Clear the input buffer
                getline(cin, name);
                cout << "Enter Roll Number (Integer): ";
                cin >> rollNumber;

```

```
cout << "Enter Class (String): ";
cin >> className;
cout << "Enter Division (Char): ";
cin >> division;
cout << "Enter Date of Birth (DD/MM/YYYY) (String): ";
cin >> dateOfBirth;
cout << "Enter Blood Group: ";
cin >> bloodGroup;
cout << "Enter Street Address (String): ";
cin.ignore();
getline(cin, street);
cout << "Enter City (String): ";
getline(cin, city);
cout << "Enter State (String): ";
getline(cin, state);
cout << "Enter Country (String): ";
getline(cin, country);
cout << "Enter Zip Code (Integer): ";
cin >> zipCode;
cout << "Enter Telephone Number (Integer): ";
cin >> telephoneNumber;
cout << "Enter Driving License Number (String): ";
cin.ignore();
getline(cin, drivingLicenseNumber);

// Create a new Student object
```

```

        students[studentCount++] = new Student(name, rollNumber,
        className, division, dateOfBirth, bloodGroup, street, city, state,
        country, zipCode, telephoneNumber, drivingLicenseNumber);

        cout << "Student added successfully!" << endl;

        break;
    }

    case 2: { // Update
        int rollNumberToUpdate;

        cout << "Enter the roll number of the student to update: ";
        cin >> rollNumberToUpdate;

        for (int i = 0; i < studentCount; i++) {
            if (students[i]->getRollNumber() == rollNumberToUpdate) {
                string newClassName;

                cout << "Enter new class name: ";

                cin.ignore();

                getline(cin, newClassName);

                students[i] -> setClassName(newClassName);

                cout << "Student information updated successfully!" <<
endl;

                break;
            }
        }

        break;
    }

    case 3: {
        int rollNumberToDelete;

        cout << "Enter the roll number of the student to delete: ";
        cin >> rollNumberToDelete;
    }
}

```

```

        for(int i=0; i<studentCount; i++) {
            if (students[i]->getRollNumber() == rollNumberToDelete) {
                delete students[i]; // Deallocation
                // Shift
                for (int j = i; j < studentCount - 1; j++)
                    students[j] = students[j+1];
                studentCount--;
                cout << "Student deleted successfully!" << endl;
                break;
            }
        }
        break;
    }
    case 4: {
        // Displaying all students
        if (studentCount == 0) cout << "No students to display." <<
endl;
        else {
            for (int i = 0; i < studentCount; i++)
                cout << "Student " << (i + 1) << ":\n" << *students[i] <<
endl;
        }
        break;
    }

    case 5: cout << "Total Students: " <<
Student::getTotalStudents() << endl; break;

    case 6: cout << "Exiting the program." << endl; break;

    default: cout << "Invalid choice! Please try again." << endl;

```

```
}

// Clear input buffer to prevent infinite loop on invalid input
if (cin.fail()) {
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
}
} while (choice != 6);
for (int i = 0; i < studentCount; i++) delete students[i];
return 0;
}
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> g++ A2.cpp -o a
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> .\a.exe
```

Menu:

1. Add Student
2. Update Student
3. Delete Student
4. Display All Students
5. Display Total Students
6. Exit

Enter your choice: 1

Enter Name (String): N

Enter Roll Number (Integer): 1

Enter Class (String): SE

Enter Division (Char): A

Enter Date of Birth (DD/MM/YYYY) (String): 01/01/2001

Enter Blood Group: A

Enter Street Address (String): 12 ST

Enter City (String): Pune

Enter State (String): MH

Enter Country (String): IN

Enter Zip Code (Integer): 1

Enter Telephone Number (Integer): 1

Enter Driving License Number (String): 1

Student added successfully!

Enter your choice: 4

Student 1:

Name: N

Roll Number: 1

Class: SE

Division: A

Date of Birth: 01/01/2001

Blood Group: A

Contact Address: 12 ST, Pune, MH, IN, 1

Telephone Number: 1

Driving License Number: 1

Enter your choice: 2

Enter the roll number of the student to update: 1

Enter new class name: TE

Student information updated successfully!

Enter your choice: 3

Enter the roll number of the student to delete: 1

Student deleted successfully!

Enter your choice: 5

Total Students: 0

Enter your choice: 6

Exiting the program.

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> |
```

```

/**
 * * Eshan Nahar

 * Imagine a publishing company which does marketing for book and
 audio cassette versions. Create a class publication that stores the
 title (a string) and price (type float) of publications. From this
 class derive two classes: book which adds a page count (type int) and
 tape which adds a playing time in minutes (type float).

 * Write a program that instantiates the book and tape class, allows
 user to enter data and displays the data members. If an exception is
 caught, replace all the data member values with zero values.

 */

```

```

#include<bits/stdc++.h>

```

```

using namespace std;

```

```

class Publication {

```

```

    protected:

```

```

        string title;

```

```

        float price;

```

```

    public:

```

```

        Publication() : title(""), price(0.0f) {} // Constructor with
 default values

```

```

        void setData(const string& t, float p) {

```

```

            title = t;

```

```

            price = p;

```

```

        }

```

```

        // Virtual because we want to override it and use inheritance

```

```

        virtual void display() const {

```

```

            cout << "Title: " << title << " Price: " << price << endl;

```

```

        }

```

```
};
```

```
class Book : public Publication {  
    private:  
        int pageCount;  
    public:  
        Book() : pageCount(0) {}  
        void setData(const string& t, float p, int pc) {  
            Publication::setData(t, p);  
            pageCount = pc;  
        }  
        void display() const override {  
            Publication::display();  
            cout << "Page Count: " << pageCount << endl;  
        }  
};
```

```
class Tape : public Publication {  
    private:  
        float playingTime;  
    public:  
        Tape() : playingTime(0.0f) {}  
        void setData(const string& t, float p, float pt) {  
            Publication::setData(t, p);  
            playingTime = pt;  
        }  
        void display() const override {
```



```

        Publication::display();
        cout << "Playing Time (Minutes): " << playingTime << endl;
    }
};

void createBook() {
    Book book;
    string title; float price; int pageCount;
    cout << "Enter Book Title: ";
    cin >> title;

    cout << "Enter Book Price: ";
    while (true) {
        cin >> price;
        if (cin.fail() || price < 0) {
            cout << "Invalid input for price. Please enter a non-negative
number: ";
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
        } else {
            break;
        }
    }

    cout << "Enter Page Count: ";
    while (true) {
        cin >> pageCount;
        if (cin.fail() || pageCount < 0) {

```

```

        cout << "Invalid input for page count. Please enter a non-
negative integer: ";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    } else {
        break;
    }
}

book.setData(title, price, pageCount);
cout << "\n Book Information:" << endl;
book.display();
}

void createTape() {
    Tape tape;
    string title; float price; float playingTime;
    cout << "Enter Tape Title: ";
    cin >> title;

    cout << "Enter Tape Price: ";
    while (true) {
        cin >> price;
        if (cin.fail() || price < 0) {
            cout << "Invalid input for price. Please enter a non-negative
number: ";
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');

```

```

    } else {
        break;
    }
}

cout << "Enter Playing Time (Minutes): ";
while (true) {
    cin >> playingTime;
    if (cin.fail() || playingTime < 0) {
        cout << "Invalid input for playing time. Please enter a non-
negative number: ";
        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
    } else {
        break;
    }
}

tape.setData(title, price, playingTime);
cout << "\n Tape Information:" << endl;
tape.display();
}

int main() {
    int choice;
    do {
        cout << "==== MENU =====" << endl;
        cout << "\t 1. Create Book" << endl;

```

```
    cout << "\t 2. Create Tape" << endl;
    cout << "\t 3. Quit" << endl;
    cout << "Enter your choice: ";
    cin >> choice;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
    switch(choice) {
        case 1:
            createBook();
            break;
        case 2:
            createTape();
            break;
        case 3:
            break;
        default:
            cout << "Invalid choice" << endl;
    }
} while(choice != 3);
return 0;
}
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> g++ A3.cpp -o a
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> .\a.exe
```

```
===== MENU =====
```

1. Create Book
2. Create Tape
3. Quit

Enter your choice: 1

Enter Book Title: B

Enter Book Price: -1

Invalid input for price. Please enter a non-negative number: 1

Enter Page Count: 100

Book Information:

Title: B Price: 1

Page Count: 100

```
===== MENU =====
```

1. Create Book
2. Create Tape
3. Quit

Enter your choice: 2

Enter Tape Title: T

Enter Tape Price: 1

Enter Playing Time (Minutes): -1

Invalid input for playing time. Please enter a non-negative number: 5

Tape Information:

Title: T Price: 1

Playing Time (Minutes): 5

```
===== MENU =====
```

1. Create Book
2. Create Tape
3. Quit

Enter your choice: 3

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> |
```

```

/**
 * * Eshan Nahar
 * Write a C++ program that creates an output file, writes
information to it, closes the file, open it again as an input file
and read the information from the file
 */

#include<bits/stdc++.h>

using namespace std;

class FileHandle {
private:
    string fileName;
public:
    FileHandle(const string& file) : fileName(file) {}
    void write() {
        ofstream outputFile(fileName);
        if(outputFile.is_open()) {
            string input;
            cout << "Enter text to write the file (Type 'exit' to
stop):\n";
            while(true) {
                getline(cin, input);
                if(input == "exit") break;
                outputFile << input << endl;
            }
            outputFile.close();
            cout << "Data written to " << fileName << " successfully.\n";

```

```

        } else cerr << "Unable to open file for writing";
    }

    void read() {
        ifstream inputFile(fileName);
        if(inputFile.is_open()) {
            string line;
            cout << "Contents of " << fileName << ":\n";
            while(getline(inputFile, line))
                cout << line << endl;
            inputFile.close();
        } else cerr << "Unable to open file for reading \n";
    }
};

```

```

void displayMenu() {
    cout << "\n Menu: \n";
    cout << "1. Write to file\n";
    cout << "2. Read from file\n";
    cout << "3. Exit\n";
    cout << "Choose an option: ";
}

```

```

int main() {
    FileHandle fh("output.txt");
    int choice;
    do {

```

```
    displayMenu();  
    cin >> choice;  
    cin.ignore();  
    switch(choice) {  
        case 1: fh.write(); break;  
        case 2: fh.read(); break;  
        case 3: cout<<"Exiting"; break;  
        default: cout<<"Invalid choice";  
    }  
} while(choice != 3);  
return 0;  
}
```



```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> g++ B4.cpp -o a
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> .\a.exe
```

Menu:

1. Write to file
2. Read from file
3. Exit

Choose an option: 1

Enter text to write the file (Type 'exit' to stop):

Hello

exit

Data written to output.txt successfully.

Menu:

1. Write to file
2. Read from file
3. Exit

Choose an option: 2

Contents of output.txt:

Hello

Menu:

1. Write to file
2. Read from file
3. Exit

Choose an option: 3

Exiting

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> |
```

```
/* * Eshan Nahar  
 * Write a function template for selection sort that inputs, sorts  
and outputs an integer array and a float array.  
*/
```

```
#include<bits/stdc++.h>
```

```
#define Cio cout <<
```

```
#define Iio cin >>
```

```
#define E endl
```

```
using namespace std;
```

```
template<class T>
```

```
void selectionSort(T a[], int n) {
```

```
    for(int i=0; i<n; i++) {
```

```
        int minIdx = i;
```

```
        for(int j=i+1; j<n; j++)
```

```
            if(a[j] < a[minIdx]) minIdx = j;
```

```
        swap(a[i], a[minIdx]);
```

```
    }
```

```
}
```

```
template<class T>
```

```
void inputArray(T a[], int n) {
```

```
    Cio "Enter " << n << " elements:\n";
```

```
    for(int i=0; i<n; i++) {
```

```
        Cio "a[" << i << "] = ";
```

```
        Iio a[i];
```

```
    }
```

```
}
```

```
template<class T>
void printArray(T a[], int n) {
    Cio "Sorted array:\n";
    for(int i=0; i<n; i++)
        Cio a[i] << " ";
    cout << endl;
}
```

```
int main() {
    int choice;
    Cio "Choose type of sorting:\n";
    Cio "1. Integer sorting\n";
    Cio "2. Float sorting\n";
    Cio "3. String sorting\n";
    Cio "Enter your choice: ";
    Iio choice;
    switch(choice) {
        case 1: {
            int n;
            Cio "Enter number of elements: ";
            Iio n;
            int* intArray = new int[n];
            inputArray(intArray, n);
            selectionSort(intArray, n);
            printArray(intArray, n);
        }
    }
}
```

```

        delete[] intArray;
        break;
    }
    case 2: {
        int n;
        Cio "Enter number of elements; ";
        Iio n;
        float* floatArray = new float[n];
        inputArray(floatArray, n);
        selectionSort(floatArray, n);
        printArray(floatArray, n);
        delete[] floatArray;
        break;
    }
    case 3: {
        int n;
        Cio "Enter number of elements: ";
        Iio n;
        string* stringArray = new string[n];
        inputArray(stringArray, n);
        selectionSort(stringArray, n);
        printArray(stringArray, n);
        delete[] stringArray;
        break;
    }
}
}
}

```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> g++ B5.cpp -o a
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> .\a.exe
Choose type of sorting:
1. Integer sorting
2. Float sorting
3. String sorting
Enter your choice: 1
Enter number of elements: 3
Enter 3 elements:
a[0] = 5
a[1] = 14
a[2] = 0
Sorted array:
0 5 14
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> .\a.exe
Choose type of sorting:
1. Integer sorting
2. Float sorting
3. String sorting
Enter your choice: 2
Enter number of elements; 2
Enter 2 elements:
a[0] = 4.25
a[1] = 0.33
Sorted array:
0.33 4.25
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> 3
3
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> .\a.exe
Choose type of sorting:
1. Integer sorting
2. Float sorting
3. String sorting
Enter your choice: 3
Enter number of elements: 3
Enter 3 elements:
a[0] = Hiowe
a[1] = erionw
a[2] = xawio
Sorted array:
Hiowe erionw xawio
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> |
```

```
/**
 * * Eshan Nahar
 * Write C++ program using STL for sorting and searching user defined
 records such as personal records (Name, DOB, Telephone number, ...)
 using vector container
 * OR
 */
```

```
#include<bits/stdc++.h>
```

```
using namespace std;
```

```
class PersonalRecord {
```

```
    private:
```

```
        string name;
```

```
        string dob;
```

```
        string phone;
```

```
    public:
```

```
        PersonalRecord(const string& name = "", const string& dob = "",
const string& phone = "") : name(name), dob(dob), phone(phone) {}
```

```
        string getName() const { return name; }
```

```
        void setName(const string& name) { this->name = name; }
```

```
        // Use this->name to distinguish from member variable and
parameter passed
```

```
        void display() const {
```

```
            cout << "Name: " << name << ", DOB: " << dob << ", Phone: " <<
phone << endl;
```

```
        }
```

```
        // Using const so it won't modify the object on which it's called
```

```

// Comparision required for sorting
bool operator<(const PersonalRecord& other) const {
    return name < other.name;
}

// Equality required for searching
bool operator==(const PersonalRecord& other) const {
    return name == other.name;
}
};

class PersonalRecordManager {
private:
    vector<PersonalRecord> records;
public:
    void addRecord() {
        string name, dob, phone;
        cout << "Enter Name: "; getline(cin, name);
        cout << "Enter DOB (YYYY-MM-DD): "; getline(cin, dob);
        cout << "Enter Phone No.: "; getline(cin, phone);
        records.emplace_back(name,dob,phone);
        cout << "Record added!\n";
    }

    void displayRecords() const {
        if(records.empty()) cout << "No records.\n";
        else

```

```

        for(const auto& record : records)
            record.display();
    }

void sortRecords() {
    sort(records.begin(), records.end());
    cout << "Records sorted by name.\n";
}

void searchRecord() const {
    string searchName;
    cout << "Enter a name to search: ";
    getline(cin, searchName);
    PersonalRecord searchRecord(searchName);
    auto it = find(records.begin(), records.end(), searchRecord);
    if(it != records.end()) {
        cout << "Record found:\n";
        it -> display();
    } else cout << "Record not found.\n";
}

void deleteRecord() {
    string deleteName;
    cout << "Enter name to delete: ";
    getline(cin, deleteName);
    for(auto it=records.begin(); it!=records.end(); ++it) {
        if(it -> getName() == deleteName) {

```



```
        records.erase(it);
        cout << "Record deleted!\n";
        return;
    }
}

cout << "Record not found.\n";
}

static void printMenu() {
    cout << "\nMenu:\n";
    cout << "1. Add Record\n";
    cout << "2. Display Records\n";
    cout << "3. Sort Records\n";
    cout << "4. Search Record\n";
    cout << "5. Delete Record\n";
    cout << "6. Exit\n";

}

};
```

```
int main() {
    PersonalRecordManager manager;
    int choice;
    PersonalRecordManager::printMenu();
    do {
        cout << "Enter your choice: ";
        cin >> choice;
```

```
cin.ignore();
switch(choice) {
    case 1: manager.addRecord(); break;
    case 2: manager.displayRecords(); break;
    case 3: manager.sortRecords(); break;
    case 4: manager.searchRecord(); break;
    case 5: manager.deleteRecord(); break;
    case 6: cout << "Exiting\n"; break;
    default: cout << "Invalid choice\n"; break;
}
} while(choice!=6);
return 0;
}
```

Windows PowerShell

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> g++ C6.cpp -o a
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> .\a.exe
```

Menu:

1. Add Record
2. Display Records
3. Sort Records
4. Search Record
5. Delete Record
6. Exit

Enter your choice: 1

Enter Name: T

Enter DOB (YYYY-MM-DD): 2024-01-01

Enter Phone No.: 1

Record added!

Enter your choice: 1

Enter Name: R

Enter DOB (YYYY-MM-DD): 2024-02-02

Enter Phone No.: 2

Record added!

Enter your choice: 2

Name: T, DOB: 2024-01-01, Phone: 1

Name: R, DOB: 2024-02-02, Phone: 2

Enter your choice: 3

Records sorted by name.

Enter your choice: 2

Name: R, DOB: 2024-02-02, Phone: 2

Name: T, DOB: 2024-01-01, Phone: 1

Enter your choice: 4

Enter a name to search: R

Record found:

Name: R, DOB: 2024-02-02, Phone: 2

Enter your choice: 5

Enter name to delete: R

Record deleted!

Enter your choice: 2

Name: T, DOB: 2024-01-01, Phone: 1

Enter your choice: 6

Exiting

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> |
```

```
/**  
 * * Eshan Nahar  
 * Write a program in C++ to use map associative container. The keys  
 will be the names of states and the values will be the populations of  
 the states. When the program runs, the user is prompted to type the  
 name of a state. The program then looks in the map, using the state  
 name as an index and returns the population of the state  
 */
```

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

```
int main() {  
    map<string, long> statePopulation;  
    statePopulation["First"] = 39538223;  
    statePopulation["Second"] = 29145505;  
    statePopulation["Third"] = 21538187;  
    statePopulation["Fourth"] = 20201249;  
    statePopulation["Fifth"] = 13002700;  
    statePopulation["Sixth"] = 12812508;  
    statePopulation["Seventh"] = 11799448;  
    statePopulation["Eighth"] = 10711908;  
    statePopulation["Ninth"] = 10439388;  
    statePopulation["Tenth"] = 10077331;  
  
    string stateName;  
    cout << "Enter name of a state: ";
```

```
getline(cin, stateName);  
auto it = statePopulation.find(stateName);  
if(it != statePopulation.end()) cout << "Population of " <<  
stateName << " is " << it->second << endl;  
else cout << "State not present\n";  
return 0;  
}
```

```
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> g++ C7.cpp -o a  
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> .\a.exe  
Enter name of a state: First  
Population of First is 39538223  
PS C:\Users\redma\OneDrive\Dump\SE CS\SE_L\Main\00PL> |
```