1. Write a program to perform basic input and output using streams (cin and cout)

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

int main() {
    int num;
    cout << "Enter a number: ";
    cin >> num;
    cout << "You entered: " << num << endl;
    return 0;
}
```

2. Create a program that reads and displays multiple lines of text using cin and cout.

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

int main() {
    string line;
    cout << "Enter multiple lines of text (enter 'exit' to quit):" << endl;
    while (true) {
        getline(cin, line);
        if (line == "exit") break;
        cout << "You entered: " << line << endl;
    }
```

```
    return 0;
}
```

3. Implement a program that uses streams to read integers from the user and display their sum.

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

int main() {
    int num1, num2;
    cout << "Enter two integers: ";
    cin >> num1 >> num2;
    cout << "Sum: " << (num1 + num2) << endl;
    return 0;
}
```

4. Write a program to input and output various data types using cin and cout.

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

int main() {
    int intVar;
    float floatVar;
    double doubleVar;
    char charVar;

    cout << "Enter an integer, float, double, and character: ";
    cin >> intVar >> floatVar >> doubleVar >> charVar;
```

```
    cout << "You entered: " << endl;
    cout << "Integer: " << intVar << endl;
    cout << "Float: " << floatVar << endl;
    cout << "Double: " << doubleVar << endl;
    cout << "Character: " << charVar << endl;

    return 0;
}
```

5. Create a program that formats output using manipulators such as setw, setprecision, and fixed.

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

int main() {
    double value = 123.456789;

    cout << "Formatted output:" << endl;
    cout << "Default: " << value << endl;
    cout << "Setprecision 2: " << setprecision(2) << value << endl;
    cout << "Fixed and setprecision 2: " << fixed << setprecision(2) << value << endl;
    cout << "Width 10: " << setw(10) << value << endl;

    return 0;
}
```

6. Implement a program that reads user input for name, age, and salary, and then displays the information using formatted output.

```
#include <iostream>

#include <iomanip>

using namespace std;


int main() {

    string name;

    int age;

    double salary;


    cout << "Enter your name: ";

    cin >> name;

    cout << "Enter your age: ";

    cin >> age;

    cout << "Enter your salary: ";

    cin >> salary;


    cout << "\nFormatted Output:" << endl;

    cout << "Name: " << name << endl;

    cout << "Age: " << age << endl;

    cout << "Salary: " << fixed << setprecision(2) << salary << endl;


    return 0;

}
```

7. Write a program to demonstrate the use of ifstream and ofstream for file input and output.

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

int main() {
    // Writing to a file
    ofstream outFile("example.txt");
    outFile << "Hello, file!" << endl;
    outFile.close();

    // Reading from a file
    ifstream inFile("example.txt");
    string content;
    getline(inFile, content);
    cout << "File content: " << content << endl;
    inFile.close();

    return 0;
}
```

8. Implement a program that reads a list of integers from a file and displays them on the console.

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

int main() {
```

```
    ifstream inFile("numbers.txt");
    int num;
    while (inFile >> num) {
        cout << num << " ";
    }
    inFile.close();
    return 0;
}
```

9. Create a program that writes a list of strings to a file.

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

int main() {
    ofstream outFile("strings.txt");
    outFile << "First String\n";
    outFile << "Second String\n";
    outFile << "Third String\n";
    outFile.close();
    return 0;
}
```

10. Write a program to demonstrate unformatted input and output using get and put functions.

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

```
int main() {
    char ch;

    cout << "Enter a character: ";
    ch = cin.get();
    cout << "You entered: ";
    cout.put(ch);
    cout << endl;

    return 0;
}
```

11. Implement a program that reads and writes characters using get and put.

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

int main() {
    char ch;

    cout << "Enter a character: ";
    ch = cin.get();

    cout << "You entered: ";
    cout.put(ch);
    cout << endl;
```

```
    return 0;
}
```

12. Create a program that uses formatted input and output to display a table of data.

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

int main() {
    cout << setw(15) << left << "Name"
         << setw(10) << "Age"
         << setw(15) << "Salary" << endl;

    cout << setw(15) << left << "John"
         << setw(10) << 30
         << setw(15) << "$3000" << endl;

    cout << setw(15) << left << "Jane"
         << setw(10) << 25
         << setw(15) << "$4000" << endl;

    return 0;
}
```

13. Write a program that uses getline to read a full line of text and display it.

```
#include <iostream>
```

```
#include <string>

using namespace std;


int main() {

    string line;


    cout << "Enter a full line of text: ";

    getline(cin, line);


    cout << "You entered: " << line << endl;


    return 0;

}
```

14. Write a program that uses manipulators to format floating-point numbers with different precisions.

```
#include <iostream>

#include <iomanip>

using namespace std;


int main() {

    double num = 3.14159265358979;


    cout << "Default precision: " << num << endl;

    cout << "Precision 3: " << setprecision(3) << num << endl;

    cout << "Precision 5: " << setprecision(5) << num << endl;
```

```
    return 0;
}
```

15. Implement a program that uses setw to align text output in columns.

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

int main() {
    cout << setw(10) << left << "Name" << setw(10) << "Age" << setw(10) << "Salary" << endl;
    cout << setw(10) << left << "John" << setw(10) << 30 << setw(10) << "$3000" << endl;
    cout << setw(10) << left << "Jane" << setw(10) << 25 << setw(10) << "$4000" << endl;

    return 0;
}
```

16. Create a program that uses manipulators to format currency and percentage values.

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

int main() {
    double price = 1234.567;
    double percentage = 0.876;

    cout << "Currency format: " << fixed << setprecision(2) << "$" << price << endl;
    cout << "Percentage format: " << fixed << setprecision(2) << percentage * 100 << "%" << endl;
```

```
    return 0;
}
```

17. Write a program to read data from a text file and display it on the console.

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

int main() {
    ifstream inFile("data.txt");
    string line;

    while (getline(inFile, line)) {
        cout << line << endl;
    }

    inFile.close();
    return 0;
}
```

18. Implement a program to write user input to a text file.

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

int main() {
```

```
    ofstream outFile("userInput.txt");
    string line;

    cout << "Enter text to save to the file (enter 'exit' to quit):" << endl;
    while (true) {
        getline(cin, line);
        if (line == "exit") break;
        outFile << line << endl;
    }

    outFile.close();
    return 0;
}
```

19. Create a program that copies the contents of one file to another using file streams.

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

int main() {
    ifstream inFile("source.txt");
    ofstream outFile("destination.txt");

    string line;
    while (getline(inFile, line)) {
        outFile << line << endl;
```

```
    }

    inFile.close();
    outFile.close();
    return 0;
}
```

20. Write a program that appends new data to an existing file.

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

int main() {
    ofstream outFile("data.txt", ios::app);
    outFile << "New data appended to file." << endl;
    outFile.close();

    return 0;
}
```

21. Write a program to read binary data from a file using ifstream.

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

int main() {
    ifstream inFile("data.bin", ios::binary);
```

```
    int num;

    inFile.read(reinterpret_cast<char*>(&num), sizeof(num));
    cout << "Binary data read from file: " << num << endl;

    inFile.close();
    return 0;
}
```

22. Implement a program to write binary data to a file using ofstream.

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

int main() {
    ofstream outFile("data.bin", ios::binary);
    int num = 1234;

    outFile.write(reinterpret_cast<const char*>(&num), sizeof(num));

    outFile.close();
    return 0;
}
```

23. Create a program that demonstrates the use of fstream for both input and output operations.

```
#include <iostream>
#include <fstream>
```

```
using namespace std;

int main() {
    fstream file("data.txt", ios::in | ios::out);

    if (!file) {
        cout << "Error opening file." << endl;
        return 1;
    }

    file << "Hello, fstream!" << endl;

    file.seekg(0, ios::beg);
    string line;
    getline(file, line);
    cout << "Read from file: " << line << endl;

    file.close();
    return 0;
}
```

24. Write a program to read and write complex data structures to a file using binary file streams.

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

```
struct Employee {
    int id;
    char name[50];
};

int main() {
    Employee emp1 = {101, "John Doe"};
    ofstream outFile("employee.dat", ios::binary);
    outFile.write(reinterpret_cast<const char*>(&emp1), sizeof(emp1));
    outFile.close();

    Employee emp2;
    ifstream inFile("employee.dat", ios::binary);
    inFile.read(reinterpret_cast<char*>(&emp2), sizeof(emp2));
    cout << "Employee ID: " << emp2.id << ", Name: " << emp2.name << endl;
    inFile.close();

    return 0;
}
```

25. Write a program to rename and delete files using the rename and remove functions.

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

int main() {
```

```
    if (rename("oldfile.txt", "newfile.txt") == 0) {
        cout << "File renamed successfully." << endl;
    } else {
        cout << "Error renaming file." << endl;
    }

    if (remove("newfile.txt") == 0) {
        cout << "File deleted successfully." << endl;
    } else {
        cout << "Error deleting file." << endl;
    }

    return 0;
}
```

26. Implement a program to create, open, and close files using file streams.

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

int main() {
    ofstream outFile("file.txt");
    if (outFile.is_open()) {
        outFile << "File created and opened successfully!" << endl;
        outFile.close();
        cout << "File closed successfully." << endl;
```

```
    } else {
        cout << "Error opening file." << endl;
    }

    return 0;
}
```

27. Create a program that uses the seekg and tellg functions to manipulate file pointers.

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

int main() {
    ifstream inFile("file.txt");
    inFile.seekg(0, ios::end);
    cout << "File size: " << inFile.tellg() << " bytes" << endl;

    inFile.close();
    return 0;
}
```

28. Write a program that uses the seekp and tellp functions to set and retrieve the put pointer position.

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

int main() {
```

```
    ofstream outFile("output.txt");
    outFile << "Hello, World!" << endl;

    outFile.seekp(0, ios::end);
    cout << "Put pointer position: " << outFile.tellp() << endl;

    outFile.close();
    return 0;
}
```

29. Write a program to open a file in different modes (read, write, append) and demonstrate their effects.

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

int main() {
    // Open file in write mode
    ofstream outFile("test.txt");
    outFile << "Writing in write mode." << endl;
    outFile.close();

    // Open file in append mode
    outFile.open("test.txt", ios::app);
    outFile << "Appending in append mode." << endl;
    outFile.close();
```

```
    // Open file in read mode
    ifstream inFile("test.txt");
    string line;
    while (getline(inFile, line)) {
        cout << line << endl;
    }
    inFile.close();

    return 0;
}
```

30. Implement a program that reads from and writes to a file in binary mode.

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

int main() {
    int num = 12345;
    ofstream outFile("binary.bin", ios::binary);
    outFile.write(reinterpret_cast<const char*>(&num), sizeof(num));
    outFile.close();

    int readNum;
    ifstream inFile("binary.bin", ios::binary);
    inFile.read(reinterpret_cast<char*>(&readNum), sizeof(readNum));
    cout << "Read from binary file: " << readNum << endl;
```

```
    inFile.close();

    return 0;
}
```

31. Create a program that demonstrates the difference between text and binary file modes.

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

int main() {
    // Text file mode
    ofstream textFile("textfile.txt");
    textFile << "This is a text file.";
    textFile.close();

    // Binary file mode
    int num = 42;
    ofstream binaryFile("binaryfile.bin", ios::binary);
    binaryFile.write(reinterpret_cast<const char*>(&num), sizeof(num));
    binaryFile.close();

    cout << "Text file and binary file written successfully." << endl;

    return 0;
}
```

32. Write a program to open a file in truncation mode and demonstrate its effect.

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

int main() {
    // Writing initial data to the file
    ofstream outFile("testfile.txt");
    outFile << "This is initial content." << endl;
    outFile.close();

    // Opening file in truncation mode
    outFile.open("testfile.txt", ios::trunc);
    outFile << "This content will overwrite the previous content." << endl;
    outFile.close();

    ifstream inFile("testfile.txt");
    string content;
    while (getline(inFile, content)) {
        cout << content << endl;
    }
    inFile.close();

    return 0;
}
```

33. Write a program to read and write binary data to a file using the read and write functions.

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

int main() {
    double data = 3.14159265358979;
    ofstream outFile("data.bin", ios::binary);
    outFile.write(reinterpret_cast<const char*>(&data), sizeof(data));
    outFile.close();

    double readData;
    ifstream inFile("data.bin", ios::binary);
    inFile.read(reinterpret_cast<char*>(&readData), sizeof(readData));
    cout << "Read from binary file: " << readData << endl;
    inFile.close();

    return 0;
}
```

34. Implement a program that uses random access to read and write data at specific positions in a binary file.

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

struct Employee {
```

```
    int id;
    char name[50];
};

int main() {
    Employee emp1 = {1, "John"};
    Employee emp2 = {2, "Jane"};

    ofstream outFile("employees.dat", ios::binary);
    outFile.write(reinterpret_cast<const char*>(&emp1), sizeof(emp1));
    outFile.write(reinterpret_cast<const char*>(&emp2), sizeof(emp2));
    outFile.close();

    Employee emp;
    ifstream inFile("employees.dat", ios::binary);
    inFile.seekg(sizeof(emp1));  // Move to the second employee record
    inFile.read(reinterpret_cast<char*>(&emp), sizeof(emp));
    cout << "Employee ID: " << emp.id << ", Name: " << emp.name << endl;
    inFile.close();

    return 0;
}
```

35. Create a program that reads and writes a structure to a binary file using random access.

```
#include <iostream>
#include <fstream>
```

```
using namespace std;

struct Person {
    int id;
    char name[50];
};

int main() {
    Person p1 = {1, "Alice"};
    Person p2 = {2, "Bob"};

    ofstream outFile("person.bin", ios::binary);
    outFile.write(reinterpret_cast<const char*>(&p1), sizeof(p1));
    outFile.write(reinterpret_cast<const char*>(&p2), sizeof(p2));
    outFile.close();

    Person p;
    ifstream inFile("person.bin", ios::binary);
    inFile.seekg(sizeof(p1));  // Go to the second record
    inFile.read(reinterpret_cast<char*>(&p), sizeof(p));
    cout << "Read from binary file: ID = " << p.id << ", Name = " << p.name << endl;
    inFile.close();

    return 0;
}
```

36. Write a program that updates specific records in a binary file using random access.

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

struct Record {
    int id;
    char name[50];
};

int main() {
    Record record1 = {1, "John"};
    Record record2 = {2, "Jane"};

    // Writing to binary file
    ofstream outFile("records.dat", ios::binary);
    outFile.write(reinterpret_cast<const char*>(&record1), sizeof(record1));
    outFile.write(reinterpret_cast<const char*>(&record2), sizeof(record2));
    outFile.close();

    // Updating second record
    Record newRecord = {2, "Updated Jane"};
    fstream file("records.dat", ios::in | ios::out | ios::binary);
    file.seekp(sizeof(record1));  // Move to the second record
    file.write(reinterpret_cast<const char*>(&newRecord), sizeof(newRecord));
```

```
    file.close();

    // Reading and displaying updated data
    ifstream inFile("records.dat", ios::binary);
    while (inFile.read(reinterpret_cast<char*>(&record1), sizeof(record1))) {
        cout << "ID: " << record1.id << ", Name: " << record1.name << endl;
    }
    inFile.close();

    return 0;
}
```

37. Implement a program that reads and displays the contents of a binary file in reverse order.

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

struct Record {
    int id;
    char name[50];
};

int main() {
    // Writing data to file
    ofstream outFile("records.dat", ios::binary);
```

```
    Record record1 = {1, "John"};
    Record record2 = {2, "Jane"};
    outFile.write(reinterpret_cast<const char*>(&record1), sizeof(record1));
    outFile.write(reinterpret_cast<const char*>(&record2), sizeof(record2));
    outFile.close();

    // Reading data in reverse
    vector<Record> records;
    ifstream inFile("records.dat", ios::binary);
    Record r;
    while (inFile.read(reinterpret_cast<char*>(&r), sizeof(r))) {
        records.push_back(r);
    }
    inFile.close();

    // Displaying in reverse
    for (int i = records.size() - 1; i >= 0; --i) {
        cout << "ID: " << records[i].id << ", Name: " << records[i].name << endl;
    }

    return 0;
}
```

38. Write a program that uses streams to read user input, process it, and write the results to a file.

```
#include <iostream>
#include <fstream>
```

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

int main() {
    string name;
    int age;

    // Taking input from the user
    cout << "Enter your name: ";
    getline(cin, name);
    cout << "Enter your age: ";
    cin >> age;

    // Writing input to file
    ofstream outFile("user_info.txt");
    outFile << "Name: " << name << endl;
    outFile << "Age: " << age << endl;
    outFile.close();

    cout << "Data has been written to the file." << endl;
    return 0;
}
```

39. Implement a program that reads a configuration file and uses its settings to control program behavior.

```
#include <iostream>
#include <fstream>
```

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

int main() {
    string configSetting;

    ifstream configFile("config.txt");
    if (configFile.is_open()) {
        getline(configFile, configSetting);
        configFile.close();

        if (configSetting == "dark_mode") {
            cout << "Dark mode activated!" << endl;
        } else {
            cout << "Light mode activated!" << endl;
        }
    } else {
        cout << "Error: Could not open config file!" << endl;
    }

    return 0;
}
```

40. Create a program that logs error messages to a file using file streams.

```
#include <iostream>
#include <fstream>
```

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

int main() {
    ofstream errorLog("error_log.txt", ios::app);

    if (!errorLog) {
        cout << "Failed to open error log file!" << endl;
        return 1;
    }

    errorLog << "Error: Unable to open database connection." << endl;
    errorLog << "Error: File not found." << endl;
    errorLog.close();

    cout << "Error messages logged successfully." << endl;
    return 0;
}
```

41. Write a program that uses file streams to create a simple text or.

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

int main() {
```

```
    string inputText;
    cout << "Enter text to write to file (type 'exit' to quit):" << endl;
    ofstream outFile("or.txt");
    while (true) {
        getline(cin, inputText);

        if (inputText == "exit") {
            break;
        }

        outFile << inputText << endl;
    }
    outFile.close();
    cout << "Text has been written to or.txt." << endl;
    return 0;
}
```

42. Implement a program that reads and processes a CSV file using file streams.

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

int main() {
    ifstream file("data.csv");
```

```
    string line, word;

    while (getline(file, line)) {
        stringstream ss(line);
        while (getline(ss, word, ',')) {
            cout << word << " ";
        }
        cout << endl;
    }

    file.close();
    return 0;
}
```

43. Create a program that uses file streams to search for a specific word in a text file and count its occurrences.

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

int main() {
    string word, line;
    int count = 0;

    cout << "Enter word to search for: ";
    cin >> word;
```

```
    ifstream inFile("textfile.txt");

    while (getline(inFile, line)) {
        size_t pos = 0;
        while ((pos = line.find(word, pos)) != string::npos) {
            count++;
            pos += word.length();
        }
    }

    inFile.close();
    cout << "The word '" << word << "' appeared " << count << " times." << endl;

    return 0;
}
```

44. Write a program that demonstrates the use of exception handling with file operations.

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

int main() {
    try {
        ifstream inFile("nonexistent.txt");
```

```
        if (!inFile) {
            throw runtime_error("File not found!");
        }

        string content;
        while (getline(inFile, content)) {
            cout << content << endl;
        }

        inFile.close();
    } catch (const runtime_error& e) {
        cout << "Error: " << e.what() << endl;
    }

    return 0;
}
```

45. Implement a program that compresses and decompresses text files using simple encoding techniques.

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

void compressFile() {
    ifstream inFile("original.txt");
    ofstream outFile("compressed.txt");
```

```
    char c;
    while (inFile.get(c)) {
        outFile.put(c + 1);  // Simple encoding (shift each char)
    }

    inFile.close();
    outFile.close();
    cout << "File compressed successfully." << endl;
}

void decompressFile() {
    ifstream inFile("compressed.txt");
    ofstream outFile("decompressed.txt");

    char c;
    while (inFile.get(c)) {
        outFile.put(c - 1);  // Reversing encoding
    }

    inFile.close();
    outFile.close();
    cout << "File decompressed successfully." << endl;
}
```

```
int main() {
    compressFile();
    decompressFile();

    return 0;
}
```

46. Create a program that uses file streams to merge the contents of multiple text files into a single file.

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

int main() {
    ifstream inFile1("file1.txt");
    ifstream inFile2("file2.txt");
    ofstream outFile("merged.txt");

    string line;

    while (getline(inFile1, line)) {
        outFile << line << endl;
    }
    while (getline(inFile2, line)) {
        outFile << line << endl;
    }
```

```
    inFile1.close();
    inFile2.close();
    outFile.close();

    cout << "Files merged successfully." << endl;

    return 0;
}
```

47. Write a program that reads and processes large data files using memory-mapped files.

```
#include <iostream>
#include <fstream>
#include <sys/mman.h>
#include <fcntl.h>
#include <unistd.h>
using namespace std;
int main() {
    int fd = open("largefile.txt", O_RDONLY);
    if (fd == -1) {
        cerr << "Error opening file." << endl;
        return 1;
    }

    off_t fileSize = lseek(fd, 0, SEEK_END);
```

```
    lseek(fd, 0, SEEK_SET);

    char* data = (char*)mmap(NULL, fileSize, PROT_READ, MAP_PRIVATE, fd, 0);
    if (data == MAP_FAILED) {
        cerr << "Error mapping file." << endl;
        close(fd);
        return 1;
    }

    cout.write(data, fileSize);
    munmap(data, fileSize);
    close(fd);

    return 0;
}
```

48. Implement a program that uses streams to perform basic encryption and decryption of text files.

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

void encryptFile() {
    ifstream inFile("plaintext.txt");
    ofstream outFile("encrypted.txt");
```

```
    char c;
    while (inFile.get(c)) {
        outFile.put(c + 3);  // Caesar cipher encryption (shift by 3)
    }

    inFile.close();
    outFile.close();
    cout << "File encrypted successfully." << endl;
}

void decryptFile() {
    ifstream inFile("encrypted.txt");
    ofstream outFile("decrypted.txt");

    char c;
    while (inFile.get(c)) {
        outFile.put(c - 3);  // Reversing Caesar cipher
    }

    inFile.close();
    outFile.close();
    cout << "File decrypted successfully." << endl;
}

int main() {
```

```
    encryptFile();
    decryptFile();

    return 0;
}
```