That's an extensive and thorough list of C++ file handling and stream operation exercises, spanning basic to advanced topics. Let's start by addressing items **30–38**, with clear, concise explanations and code samples where appropriate:

---

30. Example of Using File Streams to Copy File Contents

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

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

   if (!inFile || !outFile) {
       cerr << "Error opening files." << endl;
       return 1;
   }

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

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

---

31. Main C++ File Stream Classes

- ifstream: Input file stream (for reading).
- ofstream: Output file stream (for writing).
- fstream: File stream capable of both input and output.

32. Roles of ifstream, ofstream, and fstream

- **ifstream**: Reads from files (input stream).
- **ofstream**: Writes to files (output stream).
- **fstream**: Reads from and writes to files (input/output stream).

33. Using ifstream to Read Data from a File

```
ifstream file("data.txt");
string line;

while (getline(file, line)) {
  cout << line << endl;
}
file.close();
```

34. Using ofstream to Write Data to a File

```
ofstream file("output.txt");
file << "Hello, world!" << endl;
file.close();
```

35. fstream for Input and Output

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

int main() {
  fstream file("data.txt", ios::in | ios::out);
  if (!file) {
    cerr << "Error opening file." << endl;
    return 1;
  }
  string word;
```

```
    file >> word;
    file << "\nAppended text.";
    file.close();
    return 0;
}
```

---

36. File Management Functions in C++

These functions are used to manage files:

- remove(filename): Deletes a file.
- rename(oldname, newname): Renames a file.
- open(), close(), is_open(): Stream-based file control.

---

37. Using remove and rename Functions

```
#include <cstdio>

int main() {
    rename("oldfile.txt", "newfile.txt");
    remove("newfile.txt");
    return 0;
}
```

---

38. Purpose of seekg and seekp

- **seekg(pos)**: Moves the *get* (read) pointer to a specific position.
- **seekp(pos)**: Moves the *put* (write) pointer to a specific position.
  These allow **random access** in files.

Great! Let's cover questions **39 to 51**, with explanations and sample C++ code:

---

39. Examples of File Pointer Manipulation (seekg and seekp)

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

```
using namespace std;


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


   file.seekp(5);          // Move write pointer to 5th byte
   file << "XYZ";          // Overwrite from byte 5


   file.seekg(0);          // Move read pointer to beginning
   string word;
   file >> word;           // Read from beginning
   cout << "Read: " << word << endl;


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

---

40. What are File Modes in C++?

File modes define how a file is opened—whether for reading, writing, appending, etc. They are flags passed to the file stream constructor or open() function.

---

41. Different File Modes in C++

Common modes:

- ios::in – Open for reading
- ios::out – Open for writing
- ios::app – Append to the end
- ios::binary – Open in binary mode
- ios::ate – Start at end of file
- ios::trunc – Truncate file if it exists

---

42. Specifying File Mode When Opening a File

```
fstream file("data.txt", ios::in | ios::out | ios::app);
```

---

43. Difference Between Binary and Text File Modes

- **Text mode**: Interprets newlines and other characters (e.g., \n becomes CRLF on Windows).
- **Binary mode**: Reads/writes raw bytes without interpretation.

---

44. Opening Files in Different Modes

```
ofstream outText("text.txt"); // Text write (default)

ofstream outBin("data.bin", ios::binary); // Binary write

ifstream inText("text.txt", ios::in); // Text read

fstream inout("file.txt", ios::in | ios::out); // Read & write
```

---

45. What Are Binary Files in C++?

Binary files store data in raw binary format, preserving exact memory representation, which is more efficient and compact than text files.

---

46. Reading from and Writing to Binary Files

```
#include <iostream>

#include <fstream>

using namespace std;


int main() {

  int data = 100;


  ofstream out("file.bin", ios::binary);

  out.write(reinterpret_cast<char*>(&data), sizeof(data));

  out.close();


  int input;

  ifstream in("file.bin", ios::binary);

  in.read(reinterpret_cast<char*>(&input), sizeof(input));
```

```
    in.close();


    cout << "Read: " << input << endl;
    return 0;
}
```

---

47. What Are Random Access Files in C++?

These allow direct access to any position in the file using seekg, seekp, tellg, and tellp, without reading sequentially.

---

48. Performing Random Access Operations

```
fstream file("data.bin", ios::in | ios::out | ios::binary);
file.seekp(2 * sizeof(int)); // Move to 3rd int position
int value = 999;
file.write(reinterpret_cast<char*>(&value), sizeof(value));
file.close();
```

---

49. Example: Random Access in Binary Files

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

int main() {
    fstream file("numbers.bin", ios::in | ios::out | ios::binary);
    int nums[5] = {10, 20, 30, 40, 50};

    // Write
    file.write(reinterpret_cast<char*>(nums), sizeof(nums));

    // Update 3rd number (index 2)
    int newValue = 99;
    file.seekp(2 * sizeof(int));
    file.write(reinterpret_cast<char*>(&newValue), sizeof(newValue));


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

---

50. Basic I/O Using cin and cout

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


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

---

51. Read and Display Multiple Lines of Text

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


int main() {
    string line;
    cout << "Enter multiple lines (type 'exit' to stop):" << endl;
    while (true) {
        getline(cin, line);
        if (line == "exit") break;
        cout << "You entered: " << line << endl;
    }
    return 0;
}
```

That's a comprehensive list of C++ file and stream handling exercises! Here are brief implementations or templates for a selection of these programs to help you get started.

---

52. Sum of Integers Using Streams

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

int main() {
    int num, sum = 0;
    cout << "Enter integers (non-integer to stop): ";
    while (cin >> num) {
        sum += num;
    }
    cout << "Sum = " << sum << endl;
    return 0;
}
```

---

53. Input and Output Various Data Types

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

int main() {
    int i;
    float f;
    char c;
    string s;

    cout << "Enter an int, float, char, and string: ";
    cin >> i >> f >> c >> s;

    cout << "Int: " << i << ", Float: " << f << ", Char: " << c << ", String: " << s << endl;
    return 0;
}
```

---

54. Output Formatting with Manipulators

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

int main() {
    double pi = 3.1415926535;
    cout << fixed << setprecision(2) << setw(10) << pi << endl;
    return 0;
}
```

---

55. Formatted Output for Name, Age, Salary

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

int main() {
    string name;
    int age;
    double salary;

    cout << "Enter name, age, and salary: ";
    cin >> name >> age >> salary;

    cout << left << setw(10) << "Name" << setw(5) << "Age" << setw(10) << "Salary" << endl;
    cout << left << setw(10) << name << setw(5) << age << setw(10) << fixed << setprecision(2) <<
salary << endl;

    return 0;

}
```

Great! Below are C++ program examples for exercises **55 to 80**, grouped and organized for clarity. These programs cover **formatted output**, **file input/output**, **text vs binary files**, **file modes**, and more.

---

55. Read and Display Name, Age, Salary

```
#include <iostream>

#include <iomanip>

using namespace std;


int main() {

    string name;

    int age;

    double salary;


    cout << "Enter name, age, salary: ";

    cin >> name >> age >> salary;


    cout << left << setw(10) << "Name" << setw(5) << "Age" << setw(10) << "Salary" << endl;

    cout << left << setw(10) << name << setw(5) << age << setw(10) << fixed << setprecision(2) <<
salary << endl;


    return 0;

}
```

---

56. File Input/Output with ifstream and ofstream

```
#include <iostream>

#include <fstream>
```

```
using namespace std;

int main() {
   ofstream  outFile("sample.txt");
   outFile << "Hello File!" << endl;
   outFile.close();

   ifstream inFile("sample.txt");
   string line;
   getline(inFile, line);
   cout << "File says: " << line << endl;
   inFile.close();

   return 0;
}
```

---

57. Read Integers from File

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

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

---

58. Write Strings to File

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

int main() {
   ofstream outFile("words.txt");
   outFile << "Apple\nBanana\nCherry\n";
   outFile.close();
   return 0;
}
```

---

59. Unformatted I/O with get and put

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

int main() {
   char ch;
   cout << "Enter a character: ";
   ch = cin.get();
   cout.put(ch);
   return 0;
}
```

---

60. Read/Write Characters with get and put

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

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

```
    char ch;
    while (inFile.get(ch)) {
        cout.put(ch);
    }
    inFile.close();
    return 0;
}
```

---

61. Table with Formatted I/O

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

int main() {
    cout << left << setw(10) << "Name" << setw(5) << "Age" << endl;
    cout << left << setw(10) << "Alice" << setw(5) << 30 << endl;
    cout << left << setw(10) << "Bob" << setw(5) << 25 << endl;
    return 0;
}
```

---

62. Use getline to Read Full Line

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

int main() {
    string line;
    cout << "Enter a line: ";
    getline(cin, line);
    cout << "You entered: " << line << endl;
    return 0;
}
```

---

63. Format Floating-Point Precision

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

int main() {
   double val = 123.456789;
   cout << fixed << setprecision(2) << val << endl;
   cout << fixed << setprecision(4) << val << endl;
   return 0;
}
```

---

64. Use setw to Align Columns

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

int main() {
   cout << setw(10) << "ID" << setw(10) << "Score" << endl;
   cout << setw(10) << 1 << setw(10) << 95.6 << endl;
   cout << setw(10) << 2 << setw(10) << 88.4 << endl;
   return 0;
}
```

---

65. Format Currency and Percentages

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

```
int main() {
    double salary = 12345.6789, bonus = 0.12;
    cout << "Salary: $" << fixed << setprecision(2) << salary << endl;
    cout << "Bonus: " << fixed << setprecision(2) << bonus * 100 << "%" << endl;
    return 0;
}
```

---

66. Read from Text File

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

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

---

67. Write User Input to File

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

int main() {
    ofstream outFile("userinput.txt");
    string input;
    cout << "Enter text: ";
    getline(cin, input);
    outFile << input << endl;
    outFile.close();
    return 0;
}
```

---

68. Copy File Contents

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

int main() {
    ifstream src("source.txt");
    ofstream dest("destination.txt");
    char ch;
    while (src.get(ch)) {
        dest.put(ch);
    }
    src.close();
    dest.close();
    return 0;
}
```

---

69. Append to File

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

int main() {
    ofstream outFile("log.txt", ios::app);
    outFile << "New entry added.\n";
    outFile.close();
    return 0;
}
```

---

70. Read Binary Data

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

int main() {
    ifstream inFile("binary.dat", ios::binary);
    int num;
    inFile.read(reinterpret_cast<char*>(&num), sizeof(num));
    cout << "Read number: " << num << endl;
    inFile.close();
    return 0;
}
```

---

71. Write Binary Data

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

int main() {
    int num = 12345;
    ofstream outFile("binary.dat", ios::binary);
    outFile.write(reinterpret_cast<char*>(&num), sizeof(num));
    outFile.close();
    return 0;
}
```

72. Use fstream for Input/Output

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

int main() {
    fstream file("example.txt", ios::in | ios::out | ios::trunc);
    file << "Hello World\n";
    file.seekg(0);
    string line;
    getline(file, line);
    cout << "Read: " << line << endl;
    file.close();
    return 0;
}
```

73. Read/Write Struct to Binary File

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

struct Person {
    char name[20];
    int age;
};

int main() {
    Person p = {"Alice", 30};
    ofstream outFile("person.dat", ios::binary);
    outFile.write(reinterpret_cast<char*>(&p), sizeof(p));
    outFile.close();

    Person q;
    ifstream inFile("person.dat", ios::binary);
    inFile.read(reinterpret_cast<char*>(&q), sizeof(q));
    cout << "Name: " << q.name << ", Age: " << q.age << endl;
    inFile.close();

    return 0;
}
```

---

74. Rename and Delete Files

```
#include <cstdio>

int main() {
    rename("old.txt", "new.txt");
    remove("new.txt");
    return 0;
}
```

---

75. Create, Open, Close Files

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

int main() {
    ofstream file("sample.txt");
    file << "Created file" << endl;
    file.close();
    return 0;
}
```

---

76. seekg and tellg Example

```
#include <iostream>

#include <fstream>

using namespace std;


int main() {

   ifstream inFile("data.txt");

   inFile.seekg(5);

   cout << "Current position: " << inFile.tellg() << endl;

   char ch;

   inFile.get(ch);

   cout << "Character: " << ch << endl;

   inFile.close();

   return 0;

}
```

---

77. seekp and tellp Example

```
#include <iostream>

#include <fstream>

using namespace std;


int main() {

   ofstream outFile("example.txt");

   outFile.seekp(5);

   outFile << "Hello";

   cout << "Write position: " << outFile.tellp

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

---

78. File Modes (read, write, append)

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

int main() {
    ofstream file("mode.txt", ios::app);
    file << "Appending this line.\n";
    file.close();
    return 0;
}
```

---

79. Read/Write Binary Mode

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

int main() {
    int a = 50;
    ofstream out("bin.dat", ios::binary);
    out.write(reinterpret_cast<char*>(&a), sizeof(a));
    out.close();

    int b;
    ifstream in("bin.dat", ios::binary);
    in.read(reinterpret_cast<char*>(&b), sizeof(b));
    in.close();

    return 0;
}
```

---

80. Text vs Binary File Mode

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

int main() {
   // Text
   ofstream textFile("text.txt");
   textFile << 123 << endl;
   textFile.close();

   // Binary
   int n = 123;
   ofstream binFile("binfile.dat", ios::binary);
   binFile.write(reinterpret_cast<char*>(&n), sizeof(n));
   binFile.close();

   return 0;
}
```

Here are C++ program examples for exercises **81 to 97**, covering topics like file modes, binary operations, random access, exception handling, and simple utilities like search, log, compression, and CSV handling.

---

81. Open a File in Truncation Mode

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

int main() {
   ofstream file("truncate.txt", ios::trunc);
   file << "This overwrites any existing content.\n";
   file.close();
```

```
    return 0;
}
```

---

82. Read and Write Binary Data with read and write

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

int main() {
    int x = 100;
    ofstream out("data.bin", ios::binary);
    out.write(reinterpret_cast<char*>(&x), sizeof(x));
    out.close();

    int y;
    ifstream in("data.bin", ios::binary);
    in.read(reinterpret_cast<char*>(&y), sizeof(y));
    in.close();

    return 0;
}
```

---

83. Random Access in Binary File

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

int main() {
    fstream file("numbers.bin", ios::in | ios::out | ios::binary | ios::trunc);
    int nums[5] = {10, 20, 30, 40, 50};
    file.write(reinterpret_cast<char*>(nums), sizeof(nums));
    int value = 999;
    file.seekp(2 * sizeof(int)); // 3rd element
    file.write(reinterpret_cast<char*>(&value), sizeof(value));

    file.seekg(0);
    for (int i = 0; i < 5; i++) {
        file.read(reinterpret_cast<char*>(&value), sizeof(value));
        cout << value << " ";
    }
    file.close();
    return 0;
}
```

---

84. Read/Write Structure with Random Access

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

struct Record {
    int id;
    char name[20];
};

int main() {
    fstream file("records.dat", ios::in | ios::out | ios::binary | ios::trunc);
    Record r1 = {1, "Alice"}, r2 = {2, "Bob"}, r3 = {3, "Charlie"};

    file.write(reinterpret_cast<char*>(&r1),   sizeof(r1));
    file.write(reinterpret_cast<char*>(&r2),   sizeof(r2));
    file.write(reinterpret_cast<char*>(&r3), sizeof(r3));
    file.seekg(1 * sizeof(Record)); // read Bob
    Record temp;
    file.read(reinterpret_cast<char*>(&temp), sizeof(temp));
    cout << "Read ID: " << temp.id << ", Name: " << temp.name << endl;
    file.close();
    return 0;
}
```

---

85. Update Specific Records in Binary File

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

struct Data {
    int id;
    char name[20];
};

int main() {
    fstream file("data.dat", ios::in | ios::out | ios::binary);
    Data updated = {2, "Updated"};

    file.seekp(1 * sizeof(Data)); // update second record
    file.write(reinterpret_cast<char*>(&updated), sizeof(updated));
    file.close();
    return 0;
}
```

---

86. Display Binary File in Reverse Order

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

```
int main() {
    ifstream file("numbers.bin", ios::binary);
    file.seekg(0, ios::end);
    int size = file.tellg() / sizeof(int);

    for (int i = size - 1; i >= 0; i--) {
        file.seekg(i * sizeof(int));
        int n;
        file.read(reinterpret_cast<char*>(&n), sizeof(n));
        cout << n << " ";
    }
    file.close();
    return 0;
}
```

---

87. Read, Process, and Write Result to File

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

int main() {
    int x;
    cout << "Enter a number: ";
    cin >> x;
    x *= 2;

    ofstream file("output.txt");
    file << "Double: " << x << endl;
    file.close();
    return 0;
}
```

---

88. Read Config File to Control Behavior

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

int main() {
    ifstream file("config.txt");
    string key;
    int value;
    while (file >> key >> value) {
        if (key == "threshold") {
            cout << "Threshold set to: " << value << endl;
        }
    }
    return 0;
}
```

---

89. Log Errors to File

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

int main() {
    ofstream log("error.log", ios::app);
    log << "Error: Invalid input!" << endl;
    log.close();
    return 0;
}
```

---

90. Simple Text Editor

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

int main() {
    string line;
    ofstream file("text.txt", ios::app);
    cout << "Enter text (type END to stop):\n";
    while (getline(cin, line)) {
        if (line == "END") break;
        file << line << endl;
    }
    file.close();
    return 0;
}
```

---

91. Read and Process CSV File

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

int main() {
    ifstream file("data.csv");
    string line;
    while (getline(file, line)) {
        stringstream ss(line);
        string field;
        while (getline(ss, field, ',')) {
            cout << field << "\t";
        }
        cout << endl;
    }
    return 0;
}
```

---

92. Search for Word and Count Occurrences

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

int main() {
    ifstream file("text.txt");
    string word, search = "example";
    int count = 0;

    while (file >> word) {
        if (word == search) count++;
    }

    cout << "Occurrences of '" << search << "': " << count << endl;
    return 0;
}
```

---

93. Exception Handling with Files

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

```
int main() {
    try {
        ifstream file("nofile.txt");
        if (!file) throw runtime_error("File not found");
    } catch (exception &e) {
        cerr << "Error: " << e.what() << endl;
    }
    return 0;
}
```

---

94. Simple Compression/Decompression

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

int main() {
    ifstream in("original.txt");
    ofstream out("compressed.txt");
    char ch;
    while (in.get(ch)) {
        out.put(ch + 1); // simple Caesar cipher
    }
    in.close();
    out.close();
    return 0;
}
```

---

95. Merge Multiple Files

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

```
int main() {
   ofstream out("merged.txt");
   ifstream f1("a.txt"), f2("b.txt");
   string line;

   while (getline(f1, line)) out << line << endl;
   while (getline(f2, line)) out << line << endl;

   f1.close(); f2.close(); out.close();
   return 0;
}
```

---

96. Process Large Files (Concept: Chunk Read)

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

int main() {
   ifstream file("large.txt");
   const int bufferSize = 1024;
   char buffer[bufferSize];

   while (!file.eof()) {
      file.read(buffer, bufferSize);
      cout.write(buffer, file.gcount());
   }
   file.close();
   return 0;
}
```

---

97. Basic File Encryption/Decryption

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

int main() {
   ifstream in("plain.txt");
   ofstream out("encrypted.txt");
   char ch;

   while (in.get(ch)) {
      out.put(ch ^ 0xAA);  // XOR encryption
   }
   in.close();
   out.close();
   return 0;
}
```