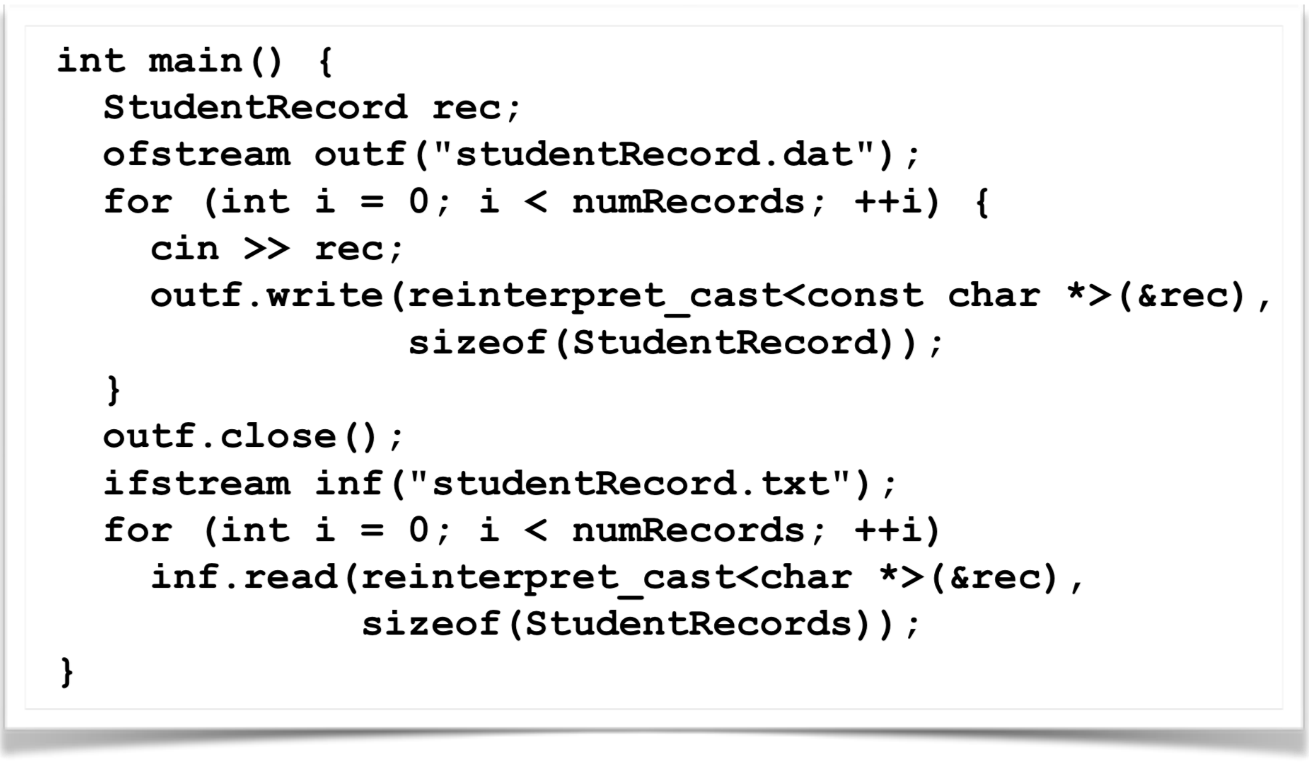

```
int main() {
  StudentRecord rec;
  ofstream outf("studentRecord.dat");
  for (int i = 0; i < numRecords; ++i) {
    cin >> rec;
    outf.write(reinterpret_cast<const char *>(&rec),
               sizeof(StudentRecord));
  }
  outf.close();
  ifstream inf("studentRecord.txt");
  for (int i = 0; i < numRecords; ++i)
    inf.read(reinterpret_cast<char *>(&rec),
             sizeof(StudentRecords));
}
```

Practice #9 (for lecture note #9)

C++ Review Part IV:

More on IO Streams

Due: 9pm, Friday, 2019-11-29

Notes and Advices:

1. Submission of the practice is NOT mandatory. The main purpose of these problems is for you to practice and get refreshed on the contents of the lecture note for the coming class meeting. Practice of the problems before class is HIGHLY recommended.
2. However, in case you need to "bargain" your term grade while you are below the next/higher letter grade for a small enough margin, you can talk to me if you have been consistently submitting your practices in time.
3. Note that your submission of the practice will NOT be graded or checked (by me or TA). Again, this is for your practice only.
4. The submissions on NTU CEIBA have strict deadline. Usually it is before midnight of the class meeting.
5. Please put all the practice problems in a directory and name it as <yourID>_pr9 (replace yourID with your ID, and omit <>) and compress it as "tar zcvf <yourID>_pr9.tgz <yourID>_pr9. Submit the .tgz file.
6. Practice problems will be, hopefully, uploaded to CEIBA no later than the previous weekend of the class.

Problems // "(pn)" refers to the page #n in the lecture note

1. (p16) Define a `class A` with a private data member "`int _data`"
   - Define a type casting member function to convert `class A` object to `int` (i.e. `return _data`).
   - Define a type casting member function to convert `class A` object to `bool` (i.e. check `(_data != 0)`). Can it co-exist with `int` convertor?
   - Define a `class B` which contains a data member "`int *_ptr`". Write a type casting member function to convert `class A` object to `B` (by setting `_ptr` as the address of `A::_data`)

   In `main()`, instantiate a `class A` object and call the above convertors to check the implementation

2. (p30) Write a "file copy" program for fun!
   - Copy an arbitrary executable file to this practice directory
   - Declare an `ifstream` object `inf` to open this executable file. Remember to read in as `ios::binary`.
   - Declare an `ofstream` object `outf` for the copied executable. Name the file as you like. Remember to read in as `ios::binary`.
   - Use "`inf.read()`" and "`outf.write()`" to read in and write out the file. Set the `streamsize` to 100. Be aware to take care of the last few bytes of the file.
   - Test if the executable has been successfully copied! You may need to "`chmod +x`" to make it executable.

   Make the input and output files as arguments of this program (Hint: see "`argc`" and "`argv`" in `main()` of homework).

   Print out some progressing message (e.g. |/-\|…) so that you can "see" that the file is being copied. You need to insert some delay on purpose to make it visible.

3. (p41) Refer to the example in p39 of the lecture note, define a class `StudentRecord` of size at least 256 Bytes. Randomly generate one million objects of this class.
   - Open a file “`studentDB.dat`” for write
   - Whenever an object is generated, write it to “`studentDB.dat`” by “`write()`” and “`reinterpret_cast`”
   - Use text editor to view ”`studentDB.dat`”. What do you see?

   Write another program to look up the ith data in “studentDB.dat”.
   - Use “`seekg()`” to position to the ith data.
   - Use “`read()`” to read in the object and `cout` it.