CENTENNIAL COLLEGE

LAB 5

COMP-217 SEC- 001

Name: Student Id:

Instructions:

* This assignment is supposed to be completed individually.
* Add you name and student Id on top of the Lab 5 Doc.
* Please add correct answer under answer section for each question.
* Submit the file with correct responses in the submission box (LAB 5) under Assignments section.
* Each question has 1 mark each.
* Please pick up the answers as a/b/c/d from options (do not add complete text).
* Please add the answers sequentially as per the questions by editing this file.
* For submission name the file with following convention: **studentfirstname\_lastname\_lab5**

Q1: Which of the following does *not* have a stream associated with it?

a. cerr.

b. cin.

c. cout.

d. All of the above have streams associated with them.

**ANS A**

Q2: In order to perform file processing in C++, which header files *must* be included?

a. <cstdio> , <iostream> and <fstream>.

b. <cstdio> and <iostream>.

c. <cstdio> and <fstream>.

d. <iostream> and <fstream>.

**ANS** **C**

Q3: Which of the following is *not* true about files?

a. C++ views each file as a sequential stream of bytes.

b. Files are opened by creating objects of stream classes.

c. Member functions of stream objects can be applied to file streams.

d. istream, ostream and iostream are derived from ifstream, ofstream and fstream, respectively.

**ANS D**

Q4: Select the *false* statement.

a. C++ imposes no structure on a file.

b. C++ files include information about their structure.

c. The programmer must impose a structure on a file.

d. C++ files do not understand notions such as “records” and “fields.”

**ANS A**

Q5: Which file open mode would be used to write data only at the end of an existing file?

a. ios::app

b. ios::in

c. ios::out

d. ios::trunc

**ANS A**

Q6: When used with ofstream objects, operator! is *not*:

a. Overloaded.

b. Used to determine if the open operation succeeded.

c. Used to close a file explicitly.

d. Used to return a nonzero value if an error occurs.

**ANS C**

Q7 [C++11]: Which of the following statements is *true*?

a. When opening a file, you can specify the name of the file only as a pointer-based string.

b. When opening a file, you can specify the name of the file only as a string object.

c. When opening a file, you can specify the name of the file as either a pointer-based string or a string object.

d. None of the above.

**ANS C**

Q8: Which of the following will *not* change the file-position pointer to the same position as the others? Assume a 10-byte file size and a current position at byte # 1.

a. fileObject.seekg(2);

b. fileObject.seekg(1, ios::cur);

c. fileObject.seekg(2, ios::beg);

d. fileObject.seekg(8, ios::end);

**ANS A**

Q9: What is *not* true about this code segment?

location = fileObject.tellg();

a. tellg is a member function of fileObject.

b. location is a pointer.

c. The value of location after the segment executes must be less than or equal to the number of bytes in the file attached to fileObject.

d. fileObject is an istream object.

**ANS B**

Q10: The new C++14 stream manipulator \_\_\_\_\_\_\_\_ enables a program to read quoted text from a stream, including any white space characters in the quoted text, and discards the double quote delimiters. For example, if we read

100 "Janie Jones" 24.98

using

inClientFile>> account>> quoted(name)>> balance;

the first stream extraction reads 100 into the int variable account, the second reads Janie Jones as one string and stores it in the string variable name without the double-quote delimiters, and the third stream extraction reads 24.98 into the double variable balance.

a. discard

b. quoted

c. hide

d. literal

**ANS: B**

Q11: Which of the following is *not* a disadvantage of trying to modify a sequential access file?

a. Modifying data can potentially destroy other data.

b. It may be necessary to rewrite every record in the file to make a slight change.

c. Things that are stored in the same number of “raw data” bytes internally may *not* take up the same amount of space in a file.

d. Overwriting a record with another record of the *same* size is very difficult.

**ANS: D**

Q12: Random access files are *more* effective than sequential files for:

a. Instant access to data.

b. Updating data easily.

c. Inserting data into the file without destroying other data.

d. All of the above.

**ANS: D**

Q13: A random access file is organized *most* like a(n):

a. Array.

b. Object.

c. Class.

d. Pointer.

**ANS: A**

Q14: Select the *false* statement. The write function:

a. Creates unformatted data.

b. Expects data type const char\* as its first argument.

c. Writes to files in hexadecimal format.

d. Takes an argument of type size\_t.

**ANS: C**

Q15: Select the *false* statement. The reinterpret\_cast operator:

a. Changes the value of the object to which its operand points.

b. Performs its operation at compile time.

c. Is compiler-dependent and can cause programs to behave differently on different platforms.

d. Is easy to use to perform dangerous manipulations that could lead to serious execution-time errors.

**ANS: B**

Q16: To write *fixed-length* records, use file open mode:

a. ios::app

b. ios::ate

c. ios::trunc

d. ios::binary

**ANS: D**

Q17: Which of the following functions would *not* be used to write data randomly to a random access file?

a. tellg

b. seekp

c. write

d. All of the above would be used.

**ANS: A**

Q18: For an ifstream object A, a class type B and a local variable of type B called C, the proper way to read in one B object from A into C is:

a. A.read(&C, sizeof(B));

b. A.read(reinterpret\_cast<char\*>(&C), sizeof(B));

c. A.read(reinterpret\_cast<char\*>(&C), B);

d. A.read(char\* (&C), sizeof(B));

**ANS: B**

Q19: Select the proper object type.

\_\_\_\_\_\_\_\_\_\_ file("file.dat", ios::in | ios::out);

a. iostream

b. fstream

c. ofstream

d. ifstream

**ANS: B**

Q20: Which C++ data type was designed to store Unicode characters?

a. char

b. long

c. wchar\_t

d. size\_t

**ANS C**

Q21 [C++11] Which of the following statements is false?

a. The C++ standard redesigned the classic C++ stream classes, which processed only chars, as class templates with specializations for processing characters of types char and wchar\_t, respectively.

b. C++11 added type char64\_t to handle the new double-width Unicode characters.

c. The size of type wchar\_t is not specified by the C++ standard.

d. C++11’s new char16\_t and char32\_t types for representing Unicode characters were added to provide character types with explicitly specified sizes.

ANS: B

Q22: Which of the following classes is a *base class* of the other three?

a. basic\_ios

b. basic\_istream

c. basic\_ostream

d. basic\_iostream

**ANS A**

Q23: Which of the following is *not* an object of the ostream class?

a. cout

b. cerr

c. cin

d. clog

**ANS C**

Q24: Which of the following classes is *deepest* in the inheritance hierarchy?

a. basic\_iostream

b. basic\_ofstream

c. basic\_ifstream

d. basic\_fstream

**ANS A**

Q25: Which of the following is *not* a member function of the C++ ostream class?

a. Stream-insertion operator (<<).

b. Stream-extraction operator (>>).

c. put.

d. write.

**ANS B**

Q26: Which of the following prints the address of character string string given the following declaration?

char\* string{"test"};

a. cout << string;

b. cout << \*&string;

c. cout << static\_cast<void\*>(string);

d. cout << \*string;

**ANS: C**

Q27: Which of the following is an *illegal* use of function put?

a. cout.put('A');

b. cout.put("A");

c. cout.put('A').put('\n');

d. cout.put(65);

**ANS: B**

Q28: The stream-extraction operator:

a. Does *not* normally accept white-space characters.

b. Returns true when the end-of-file is encountered.

c. Sets the stream’s failbit if the operation fails.

d. Sets the stream’s badbit if the data is of the wrong type.

**ANS: A**

Q29: One *difference* between the three-argument version of the get function and the getline function is that:

a. Only get has a delimiter.

b. The getline function removes the delimiter from the stream.

c. Only get adds the delimiter to the array.

d. getline stores the characters it reads into its character array argument.

**ANS: B**

Q30: The putback member function returns to the input stream the previous character obtained by:

a. A get from the input stream.

b. Using the stream extraction operator on the input stream.

c. Reading input from the keyboard.

d. Reading a file from disk.

**ANS: A**

Q31: Upon encountering the designated delimiter character, the ignore member function will:

a. Read it in and return its value.

b. Ignore it and continue reading and discarding characters.

c. Terminate.

d. Replace it with an EOF character.

**ANS: B**

Q32: If *unexpected* data is processed in an I/O operation:

a. An exception will be thrown.

b. An error message will automatically be displayed.

c. The program will terminate execution.

d. Various error bits will be set.

**ANS: D**

Q33: Which of the following is a *difference* between the read and write functions?

a. One performs formatted I/O and the other does not.

b. They take *different* types of parameters.

c. write and gcount are member functions of the same class, whereas read is not.

d. The failbit is set *only* with read.

**ANS: A**

Q34: Which of the following is *not* a difference between hex and setbase?

a. setbase is a parameterized stream manipulator and hex is not.

b. setbase is provided by a *different* header file than hex.

c. setbase(16) and hex have *different* effects on stream output.

d. setbase takes an argument but hex does not.

**ANS: D**

Q35: What will be *output* by the following statements?

double x{1.23456789};  
cout << fixed;

cout << setprecision(5) << x << endl;

cout.precision(3);

cout << x << endl;

cout << x << endl;

a. 1.2346  
1.23  
1.23

b. 1.23457  
1.235  
1.23456789

c. 1.2346  
1.23  
1.23456789

d. 1.23457  
1.235  
1.235

**ANS: C**

Q36: Which of the following is *not* true about setw and width?

a. If the width set is *not* sufficient the output prints as wide as it needs.

b. They are used to set the field width of output.

c. Both of them can perform two tasks, setting the field width and returning the current field width.

d. They only apply for the next insertion/extraction.

**ANS: A**

Q37: Which of the following is a *valid* user-defined output stream manipulator header?

a. ostream& tab(ostream& output)

b. ostream tab(ostream output)

c. istream& tab(istream output)

d. void tab(ostream& output)

**ANS: A**

Q38: What will be *output* by the following statement?

cout << showpoint << setprecision(4) << 11.0 << endl;

a. 11

b. 11.0

c. 11.00

d. 11.000

**ANS: C**

Q39: Which of the following stream manipulators causes an outputted number’s sign to be *left justified*, its magnitude to be *right justified* and the center space to be filled with fill characters?

a. left

b. right

c. internal

d. showpos

**ANS: C**

Q40: Which of the following statements restores the *default fill character*?

a. cout.defaultFill();

b. cout.fill();

c. cout.fill(0);

d. cout.fill(' ');

**ANS: D**

Q41: When the showbase flag is set:

a. The base of a number precedes it in brackets.

b. Decimal numbers are *not* output any differently.

c. "oct" or "hex" will be displayed in the output stream.

d. Octal numbers can appear in one of two ways.

**ANS: C**

Q42: What will be *output* by the following statements?

double x{.0012345};  
 cout << fixed << x << endl;

cout << scientific << x << endl;

a. 1.234500e-03  
0.001234

b. 1.23450e-03  
0.00123450

c. .001234  
1.234500e-03

d. 0.00123450  
1.23450e-03

**ANS: A**

Q43: Which of the following outputs does *not* guarantee that the uppercase flag has been set?

a. All hexadecimal numbers appear in the form 0X87.

b. All numbers written in scientific notation appear the form 6.45E+010.

c. All text outputs appear in the form SAMPLE OUTPUT.

d. All hexadecimal numbers appear in the form AF6.

**ANS: D**

Q44: Which of the following is *not* true about bool values and how they're output with the output stream?

a. The old style of representing true/false values used -1 to indicate false and 1 to indicate true.

b. A bool value outputs as 0 or 1 by default.

c. Stream manipulator boolalpha sets the output stream to display bool values as the strings "true" and "false".

d. Both boolalpha and noboolalpha are “sticky” settings.

**ANS: A**

Q45: To reset the format state of the output stream:

a. Call the reset member function.

b. Call the flags member function with the ios\_base::fmtflags constant as the argument.

c. Save a copy of the fmtflags value returned by calling member function flags *before* making any format changes, and then call flags again with that fmtflags value as the argument.

d. You must manually apply each individual format change member function or stream manipulator to restore the default format state.

**ANS: C**

Q46: The good member function will return false if:

a. The eof member function would return true.

b. The bad member function would return true.

c. The failbit member function would return true.

d. Any of the above.

**ANS: D**

Q47: The difference between the operator! member function and the operator void\* member function is that:

a. They always return opposite boolean values.

b. They occasionally return opposite boolean values.

c. Of the two member functions, only operator! checks if eof has been set.

d. Of the two member functions, only operator void\* checks if eof has been set.

**ANS: C**

Q48[C++11]: Overloaded operators can be used to test a stream’s state in conditions. The operator! member function, inherited into the stream classes from class basic\_ios, returns true if the badbit, the failbit or both are true. The \_\_\_\_\_\_\_\_ member function (added in C++11) returns false if the badbit is true, the failbit is true or both are true.

a. operator boolalpha

b. operator!

c. operator boolean

d. operator bool

ANS: D

Q49: Select the *false* statement. Outputs are:

a. Flushed automatically at the end of a program.

b. Flushed when the buffer fills.

c. Able to be synchronized with inputs.

d. Never automatically tied to inputs.

**ANS: A**

Q50: Untying an input stream, inputStream, from an output stream, outputStream, is done with the function call:

a. inputStream.untie().

b. inputStream.untie(&outputStream).

c. inputStream.tie().

d. inputStream.tie(0).

**ANS: D**