READ THIS NOW!

Failure to read and follow the instructions below may result in severe penalties.

- Print your name in the space provided below.
- Print your name and ID number on the Opscan form and code your ID number correctly on the Opscan form.
- Choose the single best answer for each question — some answers may be partially correct. If you mark more than one answer to a question, you will receive no credit for any of them.
- Unless a question involves determining whether given C++ code is syntactically correct, assume that it is. Unless a question specifically deals with compiler #include directives, you should assume the necessary header files have been included.
- Be careful to distinguish integer values from floating point values (containing a decimal point). In questions/answers that require a distinction between integer and real values, integers will be represented without a decimal point, whereas real values will have a decimal point, [1704 (integer), 1704.0 (floating point)].
- This is a closed-book, closed-notes examination.
- No laptops, calculators or other electronic devices may be used during this examination.
- You may not discuss (in any form: written, verbal or electronic) the content of this examination with any student who has not taken it.
- You must return this test form when you complete the examination. Failure to adhere to any of these restrictions is an Honor Code violation.
- There are 25 equal-valued multiple-choice questions.
- The answers you mark on the Opscan form will be considered your official answers.
- When you have finished, sign the pledge at the bottom of this page and turn in the test and your Opscan.

Do not start the test until instructed to do so!

Name (Last, First) ________________________________
printed

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

____
signature

1. One factor in the total cost of developing a large software system is the number of separate modules into which the design is decomposed. Which of the following best describes the expected relationship between the total cost and the number of modules, assuming the total amount of code required is essentially constant?

   1) The total cost increases as the number of modules decreases.
   2) The total cost decreases as the number of modules decreases.
   3) The total cost is essentially constant whether the number of modules increases or decreases.
   4) There is some "worst" number of modules, before which the total cost increases and after which the total cost decreases.
   5) None of these

---

2. When developing a software system, the cost of correcting an error depends in part on when the error is caught. Which of the following best describe(s) the expected relationship?

   1) Errors caught during design cost least to correct.
   2) Errors caught during design cost most to correct.
   3) Errors caught during coding/testing cost least to correct.
   4) Errors caught during coding/testing cost most to correct.
   5) Errors caught after deployment cost least to correct.
   6) Errors caught after deployment cost most to correct.
   7) 1 and 4 only
   8) 2 and 3 only
   9) 1 and 6 only
   10) 2 and 5 only

---

3. According to the discussion in this course, the term *information hiding* refers to:

   1) The restriction of access to data values and/or functions.
   2) The bundling of data values and the functions that can act on those values into a unit.
   3) The way that programming language code is organized in files.
   4) 1 and 2 only
   5) 1 and 3 only
   6) 2 and 3 only
   7) None of these

---

4. According to Stroustrup, the goal of adding support for a class mechanism to C++ was:

   1) to provide improved support for error detection.
   2) to provide support for restricting access to data.
   3) to provide a means of bundling data values together.
   4) All of these
   5) 1 and 2 only
   6) 1 and 3 only
   7) 2 and 3 only
   8) None of these

---

5. In object-oriented programming we often refer to *client code* or the *client* of a class. What does the term *client code* mean?

   1) Code that implements the member functions and operators of the class.
   2) Code written to use the class, but outside implementation of the class.
   3) Code written by the person who paid for the development of the class.
   4) All of these.
   5) 1 and 2 only
   6) 1 and 3 only
   7) 2 and 3 only
   8) None of these.

For the next six questions, consider the following class that will represent a linear polynomial:

```
class LinearPoly {
public:
   LinearPoly(double Slope = 0.0, double Intercept = 0.0);
   double Evaluate(double x) const;
   __________ operator+(const LinearPoly& RHS) const;
   // . . .
   bool operator==(const LinearPoly& RHS) const;
   // . . .
private:
   double m;    // slope of line
   double b;    // y-intercept of line
};
```

For the next two questions, assume the following declarations:

```
double Intercept, Slope;
LinearPoly F(3, 5);
```

6. The class doesn't provide any reporter functions for its data members. Is there any way for a client to place the value of the data member `b` of the `LinearPoly` object `F` into the variable `Intercept`?

1) Not at all.	5) 2, 3 and 4
2) Yes: `Intercept = F.b;`	6) 2 and 3 only
3) Yes: `Intercept = F(0);`	7) 2 and 4 only
4) Yes: `Intercept = F.Evaluate(0);`	8) None of these.

7. Is there any way for a client to place the value of the data member `m` of the `LinearPoly` object `F` into the variable `Slope`, assuming that the variable `Intercept` holds the value of the data member `b` of the same object?

1) Not at all.	5) 2, 3 and 4
2) Yes: `Slope = F.m;`	6) 2 and 3 only
3) Yes: `Slope = F(1) - Intercept;`	7) 2 and 4 only
4) Yes: `Slope = F.Evaluate(0) - Intercept;`	8) None of these.

8. Consider the following implementation of the `operator==` for the class `LinearPoly`:

```
bool LinearPoly::operator==(const LinearPoly& RHS) const {

     return ( m == RHS.m && b == RHS.b );
}
```

Why is it legal for the code above to directly access the private data members of `RHS`?

1) It's not.
2) Because data members of a class can always be directly accessed by other code.
3) Because the access control `private` only applies to code that's outside the implementation of the class.
4) Because the access control `private` only applies to data members that are of a `class` type, not simple types.
5) None of these.

Consider implementing the addition operator for the `LinearPoly` class:

```
___________ LinearPoly::operator+(const LinearPoly& RHS) const {  // Line 1

   double sumA, sumB;                                                 // 2
   sumA = m + RHS.m;                                                  // 3
   sumB = b + RHS.b;                                                  // 4
   return (________________);                                         // 5
}
```

9. In line 1, what does the use of `const` imply?

   1) The operator is not allowed to modify its <u>left</u> operand.
   2) The operator is not allowed to modify its <u>right</u> operand.
   3) Both 1 and 2.
   4) None of these

10. In line 1, how should the return type be specified?

   1) `void`
   2) `LinearPoly`
   3) `double`
   4) None of these

11. How should the blank in line 5 be filled?

   1) `sumA + sumB`
   2) `sumA, sumB`
   3) `LinearPoly( sumA, sumB )`
   4) `sumA`
   5) `sumB`
   6) It should be blank.
   7) None of these

For the next three questions, consider the code fragment:

```
string* pStr = NULL;                                          // Line 1
pStr = ______________;                                        //      2
_________ = "I'm the target of pStr";                         //      3
cout << "pStr points to the address:    " << ____________ << endl;  //      4
```

12. If the purpose of line 2 is to allocate a target for `pStr`, how should the blank be filled?

   1) `string`
   2) `&string`
   3) `*string`
   4) `new string`
   5) None of these

13. If the purpose of line 3 is to set the value of the target of `pStr`, how should the blank be filled?

   1) `pStr`
   2) `&pStr`
   3) `*pStr`
   4) The value cannot be set.
   5) None of these

14. If the purpose of line 4 is to print the specified address, how should the blank be filled?

   1) `pStr`
   2) `&pStr`
   3) `*pStr`
   4) The value cannot be printed.
   5) None of these

For the next four questions, consider the following short program:

```
int main() {

   ifstream File("Data.txt");                              // Line  0

   const unsigned int SZ       = 100;                      // Line  1
   const unsigned int MXCHUNKS =  10;                      // Line  2
   char  Next;                                             // Line  3
   char* Buffer = new char[SZ];                            // Line  4

   for (unsigned int Chunk = 0; Chunk < MXCHUNKS; Chunk++) { // Line  5

      for (unsigned int Pos = 0; Pos < SZ; Pos++) {        // Line  6
         File.get(Next);                                   // Line  7
         Buffer[Pos] = Next;                               // Line  8
      }
      Buffer = new char[SZ];                               // Line  9
   }

   delete [] Buffer;                                       // Line 10
   File.close();                                           // Line 11
   return 0;                                               // Line 12
}
```

Recall that a *memory leak* occurs when a program allocates memory dynamically, and then loses access to that memory without deallocating it. The design of the code fragment above causes a memory leak.

15. The memory leak can be repaired (without breaking the correctness of the rest of the code) by inserting one additional statement. What statement should be added?

    1) `delete File;`
    2) `delete Buffer;`
    3) `delete [] Buffer;`
    4) `delete double;`
    5) None of these

16. Where should the statement be added?

    1) Immediately after line 4.
    2) Immediately after line 5.
    3) Immediately before line 9.
    4) Immediately after line 9 (in loop).
    5) Immediately before line 10.
    6) Immediately after line 10.
    7) None of these

17. A `char` occupies 1 byte of memory. How many bytes of memory are "leaked" by the code fragment above?

    1) 1 byte
    2) 10 bytes
    3) 100 bytes
    4) 900 bytes
    5) 1000 bytes
    6) None of these

18. The memory leak could also be repaired by removing one statement. Which one?

    1) Line 4
    2) Line 8
    3) Line 9
    4) All of these
    5) 1 or 2 only
    6) 1 or 3 only
    7) 2 or 3 only
    8) None of these

19. Would the memory leak be eliminated if line 10 was moved to be the next-to-last statement within the outer `for` loop?

    1) Yes
    2) No

For the next four questions, a program consists of three functions, `main()`, `F()` and `G()`. As indicated below, the implementation of each function is in its own `cpp` file. Some code may be omitted, as indicated by the ellipses (…) shown in the code.

```
// main.cpp
...
#include "F.h"

int main() {

   Point Q;
   F(Q);

   return 0;
```

```
// F.cpp
...
#include "G.h"

void F(Point& P) {

   G(P);
}
```

```
// G.cpp
...

void G(Point& P) {

   int temp = P.x;
   P.x      = P.y;
   P.y      = temp;
}
```

There are also (at least) two header files, as shown below:

```
// F.h
#ifndef F_H
#define F_H

...

#endif
```

```
// G.h
#ifndef G_H
#define G_H
...
void G(Point& P);

#endif
```

The type `Point` is a `struct` type, whose details are not important to this question. It is a cardinal rule of code organization that declarations of names should be written only once, and then imported where needed by using `#include` directives. In addition, declarations should not be in scope in a file where they are not needed.

20. What should be done regarding the declaration of the function `F()`?

1) Put it in `main.cpp`
2) Put it in `F.h`
3) Put it in `G.h`
4) There's no need to do anything.
5) There's no way to get this to compile.

21. Which files should `#include` the file `G.h`?

1) `main.cpp`
2) `F.h`
3) `F.cpp`
4) `G.cpp`
5) All of them
6) 1 and 2 only
7) 1 and 3 only
8) 3 and 4 only
9) None of them
10) There's no way to get this to compile.

22. Where should the declaration of the type `Point` be placed?

1) within the file `main.cpp`
2) within the file `F.h`
3) within the file `G.h`
4) within a header `Point.h` that would be `#include`'d in <u>all</u> the other header files
5) within a header `Point.h` that would be `#include`'d in <u>all</u> the `cpp` files
6) There's no way to get this to compile.

23. What is the purpose of the `#ifndef`/`#define`/`#endif` construct used in the header files shown above?

1) It is unnecessary and should be omitted.
2) It will prevent the header file from being `#include`'d in the same scope more than once.
3) It will prevent the declarations within the header file from being included in the same scope more than once.
4) It will prevent logical errors in the code that `#include`'s the header files.
5) It is used incorrectly.

For the next two questions, consider the following class, which provides a timer that represents elapsed time in hours, minutes and seconds:

```
class Timer {
private:
   unsigned int Hrs;
   unsigned int Mins;
   unsigned int Secs;

public:
   Timer();                            // timer set to 0:00:00
   Timer(unsigned int H, unsigned int M, unsigned int S);

   unsigned int Hours() const;         // return hours field
   unsigned int Minutes() const;       // return minutes field
   unsigned int Seconds() const;       // return seconds field

   bool setHours(unsigned int H);      // set hours field
   bool setMinutes(unsigned int M);    // set minutes field
   bool setSeconds(unsigned int S);    // set seconds field

   Timer operator++();                 // advance timer by one second
   Timer operator--();                 // back timer up by one second
   void  Reset();                      // reset timer to 0:00:00

   bool operator==(const Timer& RHS) const;
   bool operator!=(const Timer& RHS) const;
   // . . . and the other four relational operators
};
```

24. The designer of the class `Timer` provided both accessor and mutator functions for all three data members. Would it have been just as good to simply make the three data members `public`?

1) Yes. If you provide both an accessor and a mutator for a data member, it's unprotected anyway.
2) No, nothing's as bad as making the data members `public`.
3) Yes, unless the accessors provide some sort of error-checking code to prevent the client from messing up.
4) Yes, unless the mutators provide some sort of error-checking code to prevent the client from messing up.
5) 3 and 4 only
6) None of these

25. If the class declaration is placed in a header file `Timer.h` and the implementations of the member functions are placed in a source file `Timer.cpp`, will it be necessary to `#include Timer.h` into `Timer.cpp`?

1) No, only client code will need to `#include` the header file.
2) Yes, because one member function may call another member function.
3) Yes, because the name `Timer` will be used in the `cpp` file.
4) 2 and 3 only
5) None of these