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?

   1) The total cost increases as the number of modules increases.
   2) The total cost decreases as the number of modules increases.
   3) There is some "optimal" number of modules, before which the total cost decreases and after which the total cost increases.
   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) 1 and 4 only
   6) 2 and 3 only
   7) None of these

---

3. An external design document, such as a structure chart, can play several roles in the development of a system. Which of the following are possible roles?

   1) As a means of advertising the end product to consumers.
   2) As a means of handling inter-module communication issues within the system development team.
   3) As a means of simplifying the long-term maintenance of the system.
   4) As a means of increasing system complexity.
   5) 1 and 2 only
   6) 2 and 3 only
   7) 2 and 4 only
   8) 3 and 4 only
   9) 2, 3 and 4 only
   10) None of these

---

4. According to the discussion in this course, the term *encapsulation* 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

---

5. 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 the creation of user-defined data types that can be used as naturally as the built-in types.
   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

For the next 6 questions, consider the following class declaration and partial implementation:

```
class Money {
private:
   int Dollars;
   int Cents;
public:
   Money();
   Money(int C);
   Money(int D, int C);
   int Value() const;
   ___ operator+(_______ Money_ RHS) const;
   ___ operator-(_______ Money_ RHS) const;
   void  Print(ostream& Out) const;
};

Money::Money() {
   Dollars = Cents = 0;
}
```

```
Money::Money(int C) {
   bool Negative = ( C < 0 );
   if ( Negative )
      C = -C;
   Cents   = C & 100;
   Dollars = C / 100;
   if ( Negative )
      Dollars = -Dollars;
}

Money::Money(int D, int C) {
   int Total = 100 * D + C;
   bool Negative = ( Total < 0 );
   if ( Negative )
      Total = -Total;
   Cents   = Total % 100;
   Dollars = Total / 100;
   if ( Negative )
      Dollars = -Dollars;
}

// other member
//      implementations . . .
```

Assume that the remaining member functions and operators have been implemented and are in scope, and any necessary header files have been included, and the following objects have been declared:

```
Money Balance, Credit(100, 0), Debit(24, 95);
```

6. Which of the following statements are syntactically legal?

1) `Balance = Credit;`
2) `Balance = Balance - Credit;`
3) `Money.Print(cout);`
4) All of these
5) 1 and 2 only
6) 1 and 3 only
7) 2 and 3 only
8) None of these

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

```
_________ Money::operator+( _____ Money_ RHS) const {  // Line 1

   Money Sum;                                          //      2
   Sum.Cents   = _________________;                    //      3
   int Carry   = Sum.Cents / 100;                      //      4
   Sum.Cents   = _________________;                    //      5
   Sum.Dollars = Dollars + RHS.Dollars + Carry;        //      6
   return ___________;                                 //      7
}
```

7. In line 1, what does the use of `const` shown above mean?

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

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

1) `void`
2) `Money`
3) `int`
4) None of these

9. How should the parameter in line 1 be specified?

1) `Money RHS`
2) `Money& RHS`
3) `const Money RHS`
4) `const Money& RHS`
5) None of these

10. How should the blank in line 3 be filled?

1) `LHS.Cents + RHS.Cents`
2) `Cents`
3) `Cents + RHS->Cents`
4) `Cents + RHS.Cents`
5) None of these

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

1) `Sum.Cents % 100`
2) `Cents % 100`
3) `Sum.Cents / 100`
4) `Cents / 100`
5) `Sum.Cents * 100`
6) `Cents * 100`
7) None of these

For the next 5 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
cout << "pStr is stored at the address: " << _____________ << endl;  //      5
cout << "pStr points to the string:     " << _____________ << endl;  //      6
```

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

15. If the purpose of line 5 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

16. If the purpose of line 6 is to print the specified string, 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 4 questions, consider the following code fragment:

```
double *pDistance;                          // Line 1
double totalDistance = 0.0;                 //      2
for (int Try = 0; Try < 1000; Try++) {      //      3
   cout << "Please enter a distance: ";     //      4
   pDistance = new double;                  //      5
   cin >> *pDdistance;                      //      6
   totalDistance += *pDistance;             //      7
}
pDistance = NULL;                           //      8
```

The design of this fragment causes a memory leak, which can be repaired (without breaking the correctness of the rest of the code) by inserting one additional statement.

17. What statement should be added?

1) `pDistance = NULL;`
2) `delete pDistance;`
3) `delete [] pDistance;`
4) `delete double;`
5) None of these

18. Where should the statement be added?

1) Immediately after line 1.
2) Immediately after line 3.
3) Immediately after line 5.
4) Immediately after line 7.
5) Immediately before line 8.
6) Immediately after line 8.
7) None of these

19. A double occupies 8 bytes of memory. How many bytes of memory are "leaked" by the code fragment above?

1) 8 bytes
2) 1000 bytes
3) 7992 bytes
4) 8000 bytes
5) 8008 bytes
6) None of these

20. The memory leak could also be repaired by relocating one statement. How?

1) Moving line 1 to immediately follow line 3.
2) Moving line 5 to immediately follow line 1.
3) Moving line 8 to immediately follow line 7.
4) All of these
5) 1 or 2 only
6) 1 or 3 only
7) 2 or 3 only
8) None of these

For the next two questions, consider the following function and calling code:

```
// In the calling function:
int* p = new int(0);
getAnInt(p);
cout << "Got: " << *p << endl;
```

```
// Function:
void getAnInt(int* pInt) {

   delete pInt;
   pInt = new int;
   cout << "Enter an integer value: ";
   cin  >> *pInt;
}
```

21. Which of the following describe the result(s) of the call to `getAnInt()` shown above?

1) The original target of the pointer `p` is deallocated.
2) The pointer `p` is given a new target.
3) The original target of the pointer `p` is "leaked".
4) 1 and 2 only
5) 2 and 3 only
6) None of these

22. What single change to the implementation of the function `getAnInt()` would fix the logic error it contains?

1) There is no logic error.
2) Omit the call to `delete`.
3) Omit the call to `new`.
4) Reverse the calls to `delete` and `new`.
5) Name the parameter `p`.
6) Change the parameter to: `const int* pInt`
7) Change the parameter to: `int* const pInt`
8) Change the parameter to: `int*& pInt`
9) Either 2 or 3 will do.
10) None of these.

For the next 3 questions, assume the declaration of the class Money given earlier is in scope and consider the following code fragment:

```
const int MAX = 100;
Money* Transaction[MAX];              // array of pointers
Money* Current = &Transaction[0];

// Assume code here that creates 100 Money objects and stores pointers to
// them in the array.
```

23. Which of the following statements would call the function `Value()` on the `Money` object corresponding to the index 0?

1) `Transaction[0].Value();`
2) `Transaction[0]->Value();`
3) `Transaction->Value()`
4) All of these
5) 1 and 2 only
6) 1 and 3 only
7) 2 and 3 only
8) None of these

24. Which of the following code fragments would correctly deallocate all the memory that is allocated dynamically in the code above?

1) `delete Transaction;`

2) `delete [] Transaction;`

3) `Transaction = NULL;`

4) `for (int Idx = 0; Idx < MAX; Idx++)`
   `Transaction[Idx] = NULL;`

5) `for (int Idx = 0; Idx < MAX; Idx++)`
   `delete Transaction[Idx];`

6) None of these

25. Suppose that the address of `Transaction[0]` is `1000` (in hex). What will be the value of `Current` after executing the statement: `Current++;`

1) `1000`
2) `1001`
3) `1004`
4) `1008`
5) `100C`
6) None of these