



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, it should be; if you think there is a syntax error where there should not be, ask. 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, [1044 (integer), 1044.0 (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

For the next two questions, consider executing the following code fragment (assume any additional declarations, etc, needed to make the code syntactically correct):

```
int j = 5;
while (j <= 30) {
    j = 2 * j;
    cout << j << endl;
}
```

1. How many times will the body of the loop be executed?

- | | | | |
|------|------|------------------|------------------|
| 1) 0 | 3) 2 | 5) 4 | 7) None of these |
| 2) 1 | 4) 3 | 6) infinite loop | |

2. What is the last value printed?

- | | | | |
|-----------------------|-------|-------|------------------|
| 1) Nothing is printed | 3) 10 | 5) 20 | 7) 40 |
| 2) 5 | 4) 15 | 6) 30 | 8) None of these |

3. A function, G, has two formal parameters, P1 of type `int` and P2 of type `char`. The data flow (communication) for variable P1 is one-way, into and out of the function. The data flow for variable P2 is one-way, out of the function. Which of the following is the **most appropriate** prototype for G?

- | | |
|--|------------------|
| 1) <code>void G(int P1, char P2);</code> | 6) 1 and 2 only |
| 2) <code>void G(int P1, char& P2);</code> | 7) 3 and 4 only |
| 3) <code>void G(int& P1, char P2);</code> | 8) 1 and 3 only |
| 4) <code>void G(int& P1, char& P2);</code> | 9) None of these |
| 5) All of them are equally appropriate. | |

4. In C++, struct variables differ from arrays in what way(s)?

- | | |
|--|------------------|
| 1) whether the elements must be of the same type | 5) 1 and 2 only |
| 2) whether the elements are accessed by location | 6) 1 and 3 only |
| 3) whether aggregate assignment is supported | 7) 2 and 3 only |
| 4) All of them | 8) None of these |

5. Consider the following function:

```
int H(int i, int j) {
    int Sum = 0;
    for ( ; i > 0; i = i - 2) {
        for (int k = 0; k <= j; k = k + 2)
            Sum = Sum + i + k;
    }
    return Sum;
}
```

What value is returned from the call `H(5, 4)`?

- | | | | |
|-------|-------|-------|------------------|
| 1) 0 | 3) 24 | 5) 32 | 7) 45 |
| 2) 14 | 4) 30 | 6) 42 | 8) None of these |

For the next three questions, consider execution of the following program:

```
void Foo(int Harry, int& Potter);
int Tmp = 3;
int main() {
    int Alpha = 12, Beta = 2;
    Foo(Alpha, Beta);
    cout << "Alpha = " << Alpha << endl;
    cout << "Beta = " << Beta << endl;
    cout << "Tmp = " << Tmp << endl;
    return 0;
}

void Foo(int Harry, int& Potter) {
    int Tmp;
    Tmp = Harry;
    Harry = Harry - 2 * Potter;
    Potter = Tmp;
}
```

6. What value is printed for the variable Tmp?

- | | | | |
|-------|------|-------|------------------|
| 1) -2 | 3) 3 | 5) 12 | 7) 21 |
| 2) 2 | 4) 8 | 6) 16 | 8) None of these |

7. What value is printed for the variable Alpha?

- | | | | |
|-------|------|-------|------------------|
| 1) -2 | 3) 3 | 5) 12 | 7) 21 |
| 2) 2 | 4) 8 | 6) 16 | 8) None of these |

8. What value is printed for the variable Beta?

- | | | | |
|-------|------|-------|------------------|
| 1) -2 | 3) 3 | 5) 12 | 7) 21 |
| 2) 2 | 4) 8 | 6) 16 | 8) None of these |

9. Given the following code fragment:

```
struct Name {
    string First;
    string Last;
};

struct Student {
    Name StudentName;
    string ID;
    double FinalAverage;
};

Student MyStudent;
```

which of the following statements are legal?

- | | |
|--|-------------------|
| 1) MyStudent.StudentName.First = "Horace"; | 5) 1 and 2 only |
| 2) MyStudent.First = "Virgil"; | 6) 1 and 3 only |
| 3) StudentName.Last = "Seneca"; | 7) 2 and 3 only |
| 4) All are legal | 8) None are legal |

10. Assume the following declarations:

```
void Fix(double realParameter, int& intParameter);
int Count = 12;
double Area = 6.28;
```

Which of the following would represent logically and syntactically appropriate call of the function `Fix`?

- | | |
|---|---------------------------------|
| 1) <code>Fix(6.85, 24);</code> | 6) All of them are appropriate. |
| 2) <code>Fix(6.85, Count);</code> | 7) 1 and 2 only |
| 3) <code>Fix(Area, 24);</code> | 8) 2 and 5 only |
| 4) <code>Fix(Area, Count + 5);</code> | 9) 2, 4 and 5 only |
| 5) <code>Fix(Area + 2.0, Count);</code> | 10) None of these |

For the next two questions, assume the declarations below:

```
MAXSIZE = 20;
char myArray[2 * MAXSIZE + 1];
```

11. How should the blank preceding `MAXSIZE` be filled?

- | | | |
|----------------------------|---------------------------|------------------|
| 1) <code>char</code> | 4) <code>const int</code> | 7) 2 or 4 only |
| 2) <code>int</code> | 5) All of them | 8) None of these |
| 3) <code>const char</code> | 6) 1 or 3 only | |

12. What is the range of valid index values for `myArray[]`?

- | | | |
|-----------------|-----------------|------------------|
| 1) 1 through 40 | 4) 0 through 40 | 7) None of these |
| 2) 1 through 41 | 5) 0 through 41 | |
| 3) 1 through 42 | 6) 0 through 42 | |

13. Suppose the first few lines of a function are as follows:

```
string Punctuate(string FName, string LName) {
    string FullName = LName + COMMA + FName;
    . . .
```

Assuming the program compiles, the identifier `COMMA` must be:

- | | |
|--|------------------------|
| 1) local to the function <code>Calculate</code> | 5) an actual parameter |
| 2) local to the function that calls <code>Calculate</code> | 6) 1 and 4 only |
| 3) declared globally | 7) 2 and 5 only |
| 4) a formal parameter | 8) None of these |

14. Given the following type definition and variable declaration:

```
struct Part {
    string Name;
    string Make;
    string Model;
    double Price;
};

Part ComputerPart;
```

which of the following are syntactically legal statements?

- | | |
|----------------------------|---------------------|
| 1) Part.Name = "CPU"; | 6) 2 and 3 only |
| 2) void F(Part CD); | 7) 1, 2 and 3 only |
| 3) void F(const Part& HD); | 8) 2, 3, and 4 only |
| 4) void F(const Part& HD); | 9) None of these |
| 5) All are legal | |

For the next two questions, consider the following code fragment:

```
struct Name {
    string FName;
    string LName;
};
const int Size = 50;
. . .
int main() {
    Name List[Size];
    Name Staff;
    . . .
}
```

15. Which of the following are logically and syntactically valid function invocations for a function with the prototype:

```
void readName(Name& newPerson);
```

- | | | |
|------------------------|---------------------------|--------------------|
| 1) readName(List); | 4) readName(Staff); | 7) 2 and 4 only |
| 2) readName(List[3]); | 5) All of them are valid. | 8) 2, 3 and 4 only |
| 3) readName(List[50]); | 6) 2 and 3 only | 9) None of these |

16. Which of the following are valid function invocations for a function with the prototype:

```
void writeStaff(Name Employee[]);
```

- | | | |
|-------------------------|---------------------------|--------------------|
| 1) writeStaff(List); | 4) writeStaff(List[]); | 7) 1, 2 and 3 only |
| 2) writeStaff(List[3]); | 5) All of them are valid. | 8) 2, 3 and 4 only |
| 3) writeStaff(Staff); | 6) 1 and 2 only | 9) None of these |

For the next six questions, consider the incomplete function definition given below:

```
// Reverse takes an array of integers and reverses the order of the elements.
//
// Parameters:
//     List[]   array to be reversed
//     Usage    number of values stored in List[]
//
void Reverse(_____ List[], int Usage) { // Line 1
    int Left  = 0,                          // Line 2
        Right = _____;                 // Line 3
    while ( Left _____ Right ) {       // Line 4
        int Temp    = _____;          // Line 5
        List[Right] = List[Left];          // Line 6
        List[Left]  = _____;          // Line 7
        _____;                          // Line 8
        Right--;                                // Line 9
    }
}
```

17. How should the blank preceding the first parameter in line 1 be filled?

- | | | |
|--------------|---------------|-----------------------------|
| 1) const int | 3) int& | 5) It should be left blank. |
| 2) int | 4) const int& | 6) None of these |

18. How should the blank in line 3 be filled to properly initialize Right?

- | | | |
|--------------|--------------|-----------------------------|
| 1) Usage | 3) Usage - 1 | 5) It should be left blank. |
| 2) Usage + 1 | 4) 0 | 6) None of these |

19. How should the blank in line 4 be filled to properly terminate the loop?

- | | | |
|-------|-------|------------------|
| 1) < | 3) > | 5) == |
| 2) <= | 4) >= | 6) None of these |

20. How should the blank in line 5 be filled to contribute toward reversing the array elements?

- | | | |
|---------------|----------------|-----------------------------|
| 1) Left | 3) Right | 5) It should be left blank. |
| 2) List[Left] | 4) List[Right] | 6) None of these |

21. How should the blank in line 7 be filled to contribute toward reversing the array elements?

- | | | |
|----------------|---------|-----------------------------|
| 1) Right | 3) Left | 5) It should be left blank. |
| 2) List[Right] | 4) Temp | 6) None of these |

22. How should the blank in line 8 be filled?

- | | | |
|---------------------|-----------------------|-----------------------------|
| 1) Left = Right - 1 | 3) Left-- | 5) It should be left blank. |
| 2) Left++ | 4) (Left + Right) / 2 | 6) None of these |

For questions 23 through 25 assume the following declarations:

```
enum WoodSpecies {ASH, CEDAR, OAK, PINE, POPLAR, WALNUT};
```

```
struct Size {
    int Length;
    int Width;
    int Thickness;
};
```

```
struct Board {
    Size      Dimensions;
    WoodSpecies Kind;
    int      smoothFaces;
};
```

```
Board Plank, Slab;
```

23. The number of smooth faces of Plank could be set by the statement(s):

- | | |
|---------------------------|------------------|
| 1) smoothSurfaces = 2; | 5) 1 and 2 only |
| 2) Wood.smoothFaces = 4; | 6) 1 and 3 only |
| 3) Plank.smoothFaces = 0; | 7) 2 and 3 only |
| 4) All of these | 8) None of these |

24. The width of Plank could be set by the statement(s):

- | | |
|--------------------------------|------------------|
| 1) Plank.Size.Width = 12; | 5) 1 and 2 only |
| 2) Plank.Dimensions.Width = 8; | 6) 1 and 3 only |
| 3) Plank.Width = 6; | 7) 2 and 3 only |
| 4) All of the above | 8) None of these |

25. Assuming Plank has been initialized, the data in Plank could be copied into Slab by:

- | | |
|--|------------------|
| 1) Slab.Size = Plank.Size;
Slab.Kind = Plank.Kind;
Slab.smoothFaces = Plank.smoothFaces; | 4) All of them |
| 2) Slab.Size.Length = Plank.Size.Length;
Slab.Size.Width = Plank.Size.Width;
Slab.Size.Thickness = Plank.Size.Thickness;
Slab.Kind = Plank.Kind;
Slab.smoothFaces = Plank.smoothFaces; | 5) 1 and 2 only |
| 3) Slab = Plank; | 6) 1 and 3 only |
| | 7) 2 and 3 only |
| | 8) None of these |