Name:  

Instructions: Give the best possible answer to each question. All statements are guaranteed to be syntactically valid C++, unless the question is specifically about syntax. If a question asks for a value and there is not enough information to determine that value, say so.

Given the statements:

```c++
double x = 4.3;
double* ptr = &x;
```

1. The value of `ptr` is:
2. The value of `*ptr` is:
3. What does executing the following statement do?   *ptr = 24.9;

Given the declarations:

```c++
double x = 3.14, y = 2.72;
double *ptr1, ptr2;
ptr1 = &x;
ptr2 = &y;
```

Determine the effect of each of the following assignments; choose from the answers:

A. `ptr1` points to same variable as `ptr2`
B. `ptr1` and `ptr2` point to different variables with the same value
C. statement is logically invalid
D. statement is valid but has another effect

4. `ptr1 = ptr2;`
5. `*ptr1 = ptr2;`
6. `*ptr1 = *ptr2;`

Given the declarations:

```c++
struct InventoryType {
    int Quantity;
    float itemCost;
};
InventoryType Item1 = {17, 4.99};
InventoryType* pointer = &Item1;
```
Which of the following expressions correctly reference(s) the `itemCost` data member of `Item1`?

7. (`*pointer`).itemCost

8. `*pointer.itemCost`

9. `ptr -> itemCost`

Assume the C++ statement that follows is syntactically correct within its scope:

```cpp
*alpha[i].beta = 13.5;
```

Determine whether each of the following statements is true or false:

10. `alpha` is a pointer variable

11. `i` is a pointer variable

12. `beta` is a pointer member of a `struct` type variable

13. `alpha` is an array of pointer variables

14. Assuming the declarations given above for questions 7–9, what does the following statement do? Check all that apply.

```cpp
pointer = new InventoryType;
```

_____ allocates a dynamic variable named `pointer`

_____ allocates a dynamic variable and stores its address into `pointer`

15. Which of the following are potential hazards of executing the C++ `delete` operator? Check all that apply.

_____ a dangling pointer

_____ an inaccessible object
16. Which of the following are potential hazards of executing the C++ \texttt{new} operator? Check all that apply.

- a dangling pointer
- an inaccessible object

---

Given the declarations:

\begin{verbatim}
const int NumQ = 10;
int Score[NumQ] = {87, 74, 93, 89, 54, 98, 88, 34, 81, 90};
int* Accessor;
\end{verbatim}

What value(s) would be printed by each of the following code fragments?

17. \texttt{Accessor = Score;}
\texttt{cout << *Accessor;}

18. \texttt{Accessor = Score;}
\texttt{cout << Accessor[5];}

19. \texttt{Accessor = Score[5];}
\texttt{cout << *Accessor;}

20. \texttt{Accessor = Score[5];}
\texttt{cout << Accessor;}

---

Consider the following program fragment:

\begin{verbatim}
void main() {
    const int MaxCount = 5;
    int Count[MaxCount] = {1000, 1010, 1020, 1030, 1040};
    double xValue[MaxCount] = {3.14, 2.718, 1.414, 1.732, 1.1235};
    int* intPointer;
    double* dblPointer;

    cout << "address Count[0]: " << &Count[0] << endl;
    cout << "address Count[1]: " << &Count[1] << endl;
    cout << "address Count[2]: " << &Count[2] << endl;
    cout << "address Count[3]: " << &Count[3] << endl;
    cout << "address Count[4]: " << &Count[4] << endl;
    ...
}
\end{verbatim}

Suppose that execution of this program produces the following output (addresses are in hexadecimal):

\begin{verbatim}
address Count[0]: 0x0065FDB4
address Count[1]: 0x0065FDB8
address Count[2]: 0x0065FDBC
address Count[3]: 0x0065FDC0
address Count[4]: 0x0065FDC4
\end{verbatim}

Assume also that this program is executed on a system using byte-addressing, with 32-bit \texttt{ints} and 64-bit \texttt{doubles}.

---

Due: at class, Friday October 9
What value(s) would be printed by each of the following code fragments?

21. `intPointer = Count;`
   `cout << " intPointer == " << intPointer << endl;`
   `cout << "*intPointer == " << *intPointer << endl;`

22. `intPointer = &Count[3];`
   `cout << " intPointer == " << intPointer << endl;`
   `cout << "*intPointer == " << *intPointer << endl;`

23. `intPointer = Count;`
    `intPointer++;`
    `cout << " intPointer == " << intPointer << endl;`

24. `dblPointer = (double*) &xValue[0];`
    `cout << "*dblPointer == " << *dblPointer << endl;`

25. Here, assume the base address of the array `xValue` is 0x0065FDC8.
    `dblPointer = xValue;`
    `cout << " dblPointer == " << dblPointer << endl;`
    `cout << "*dblPointer == " << *dblPointer << endl;`