Advanced C++ Standard Conversions Exam

Instructions:

- Predict the output/behavior of each code snippet
- Assume 32/64-bit system where `int` is 4 bytes, `short` is 2 bytes
- Some examples may produce undefined behavior, compiler errors, or surprising results
- Duration: 120 minutes

Section 1: Integral Promotions & Conversions

1.
```cpp
double d = 1.9;
int i = d;
std::cout << i;
```

2.
```cpp
int i = -1;
unsigned int u = i;
std::cout << u;
```

3.
```cpp
long l = 2147483648;
int i = l;
std::cout << i;
```

4.
```cpp
short s = -1;
unsigned short us = s;
std::cout << us;
```

5.
```cpp
char c = '\xFF';
int i = c;
std::cout << i;
```

Section 2: Floating-Point Conversions

6.
```
float f = 0.1f;
double d = f;
std::cout << (d == 0.1);
```

7.
```
double d = 1.0e300;
float f = d;
std::cout << f;
```

8.
```
double d = std::numeric_limits<double>::quiet_NaN();
int i = d;
std::cout << i;
```

9.
```
float f = std::numeric_limits<float>::denorm_min();
double d = f;
std::cout << (d == f);
```

10.
```
float f = 0.0f;
bool b = f;
std::cout << b;
```

Section 3: Pointer & Reference Conversions

11.
```
class Base {};
class Derived : private Base {};
Derived d;
Base* bp = &d;
```

Does this compile?

12.
```
int x = 42;
void* vp = &x;
double* dp = (double*)vp;
std::cout << *dp;
```

13.
```
int* p1 = 0;
int* p2 = NULL;
int* p3 = nullptr;
std::cout << (p1 == p2 && p2 == p3);
```

14.
```
void (*funcPtr)() = nullptr;
void* vp = funcPtr;
```

Does this compile?

15.
```
const int& r = 42;
int&& rr = 42;
std::cout << r << rr;
```

Section 4: Arithmetic Conversions

16.
```cpp
unsigned short us = 1;
int i = -1;
std::cout << (us + i);
```

17.
```cpp
unsigned int u = 0;
std::cout << (u - 1);
```

18.
```cpp
float f = 1.0f;
double d = 2.0;
auto r = f + d;
std::cout << typeid(r).name();
```

19.
```cpp
long double ld = 1.0;
double d = 2.0;
auto r = ld + d;
std::cout << typeid(r).name();
```

20.
```cpp
bool b = true;
int i = b + 10;
std::cout << i;
```

Section 5: Advanced Scenarios

21.
```cpp
class A { public: int x; };
class B : public A { public: int y; };
int A::*ptr = &A::x;
int B::*bptr = ptr;
```

Does this compile?

22.
```cpp
class A {};
class B : public A {};
class C : public A {};
class D : public B, public C {};
D d;
A* a = &d;
```

Does this compile?

23.
```cpp
const int x = 10;
int* p = const_cast<int*>(&x);
*p = 20;
std::cout << x;
```

24.
```cpp
int x = 42;
float* f = reinterpret_cast<float*>(&x);
std::cout << *f;
```

25.
```cpp
class A { public: operator int() { return 42; } };
A a;
int i = a;
std::cout << i;
```

Section 6: Extreme Edge Cases

26.
```cpp
int* p = nullptr;
p++;
std::cout << p;
```

27.
```cpp
void func() {}
auto& fref = func;
```

Does this compile?

28.
```cpp
enum class E : char { X = 127 };
E e = E::X;
int i = static_cast<int>(e);
std::cout << i;
```

29.
```cpp
auto lambda = []{};
void (*fp)() = lambda;
```

Does this compile?

30.
```cpp
struct S { int x : 3; };
S s{7};
int i = s.x;
std::cout << i;
```

Final Questions (Tricky!)

31.
```cpp
bool b = nullptr;
std::cout << b;
```

32.
```cpp
int x;
int* p1 = &x;
int* p2 = p1 + 1;
std::cout << (p2 > p1);
```

33.
```cpp
template<typename T>
void f(T t) { std::cout << typeid(t).name(); }
int arr[5];
f(arr);
```

34.
```cpp
void f(int) { std::cout << "int"; }
void f(double) { std::cout << "double"; }
f('A');
```

35.
```cpp
int x = 42;
using T = int&&;
T& r = x;
std::cout << r;
```

36.
```cpp
volatile const int x = 10;
int* p = const_cast<int*>(&x);
*p = 20;
std::cout << x;
```

37.
```cpp
class C { public: void f() {} };
void (C::*ptr)() = &C::f;
std::cout << (ptr ? "valid" : "null");
```

38.
```cpp
float f1 = 0.1f;
float f2 = 0.1;
std::cout << (f1 == f2);
```

39.
```cpp
std::byte b{42};
int i = std::to_integer<int>(b);
std::cout << i;
```

40.
```cpp
auto lambda = [x = 1](){ return x; };
std::cout << sizeof(lambda);
```