Q-18

Due No due date **Points** 15 **Questions** 15 **Time Limit** 30 Minutes **Allowed Attempts** Unlimited

Take the Quiz Again

Attempt History

	Attempt	Time	Score
KEPT	Attempt 4	14 minutes	15 out of 15
LATEST	Attempt 4	14 minutes	15 out of 15
	Attempt 3	12 minutes	14 out of 15
	Attempt 2	16 minutes	15 out of 15
	Attempt 1	21 minutes	14 out of 15

Correct answers are hidden.

Submitted Jul 22 at 11:23pm

Question 1

1 / 1 pts

What is printed when you run this code?

```
int num = 0;
int *ptr = &num;
num = 5;
*ptr += 5;
cout << num << " " << *ptr << endl;
```

- Undefined; none of these
- **10 10**
- 5 10
- 5 5
- 10 5

Question 2

1 / 1 pts

These pointers should point to "nothing". Which is not correctly initialized?

- Star *ps = NULL;
- double *pd = 0;;
- **vector<int> *vp;**
- All are correctly initialized to point to nothing.
- int *pi = nullptr;

Question 3

1 / 1 pts

All of these are legal C++ statements; which of them uses the C++ ***dereferencing operator***?

```
int a = 3, b = 4;
```

- int *p = &b;
- None of these use the dereferencing operator.
- int y = a * b;
- **int x = *p;**
- z *= a;

Question 4

1 / 1 pts

What is printed when you run this code?

```
int n{};
int *p;
*p = &n;
cout << *p << endl;
```

- None of these
- No compilation errors, but undefined behavior when run
- The address value where ***n*** is stored

- Will not compile
- The value 0 (stored in n)

Question 5

1 / 1 pts

What is true about this code?

```
int n{500};
int *p = &n;
```

- &p represents the indirect value of n
- &p is the direct or explicit value of n
- p stores the same value as n
- &n is the indirect value of p
- *p is the value of n

Question 6

1 / 1 pts

Which of these is the preferred way to initialize a pointer so that it points to "nothing"?

- `vector<int> *vp(NULL);`
- `Star *ps = NULL;`
- `double *pd = 0;`
- All are equally preferred.
- `int *pi = nullptr;`

Question 7

1 / 1 pts

Assume that ***ppi*** correctly points to ***pi***. Which line prints the address of ***ppi***?

```
int main()
{
    double pi = 3.14159;
    double *ppi;
    // code goes here
    // code goes here
}
```

- `cout << *ppi;`
- `cout << &ppi;`
- `cout << &pi;`
- `cout << ppi;`
- None of these

Question 8

1 / 1 pts

What is printed when you run this code?

```
int *p = &0;
cout << *p << endl;
```

- The address value where *p* is stored
- No output; compiler error.
- No compilation errors, but undefined behavior
- The address value `0`
- The word "`nullptr`"

Question 9

1 / 1 pts

Assume that ***ppi*** correctly points to ***pi***. Which line prints the ***size*** (in bytes) of ***pi***?

```
int main()
{
    double pi = 3.14159;
    double *ppi;
```

```
    // code goes here
    // code goes here
}
```

- cout << sizeof(*pi);
- None of these
- cout << sizeof(&ppi);
- cout << sizeof(ppi);
- cout << sizeof(*ppi);

Question 10

1 / 1 pts

What is printed when you run this code?

```
int n{};
int *p = &n;
*p = 10;
n = 20;
cout << *p << endl;
```

- The address of n
- 10
- 0
- None of these
- 20

Question 11

1 / 1 pts

What is true about an uninitialized pointer?

- None of these are true
- Dereferencing it will cause a program crash
- Dereferencing it is undefined behavior
- It is set to the nullptr value
- Dereferencing it is safe, but has no effect.

Question 12

1 / 1 pts

Examine the following code. What is stored in ***a*** after it runs.

```
int f(int * p, int x)
{
    *p = x * 2;
    return x / 2;
}
. . .
int a = 3, b, c;
c = f(&b, a);
```

- 2
- 6
- 3
- Does not compile
- 1

Question 13

1 / 1 pts

Here is a fragment of pseudocode for the ***negative()*** function from your homework. What statement represents the underlined portion of code?

```
Let p point to beginning of the image
Let end be pixel one past the end of the image
While p != end
    Invert the red component
    Move p to next component
```

- *p = p + 1;
- *p++;

- (●) `p++;`
- ( ) None of these
- ( ) `&p++;`

Question 14

1 / 1 pts

Examine the following code. What is stored in ***c*** after it runs.

```
int f(int * p, int x)
{
    *p = x * 2;
    return x / 2;
}
. . .
int a = 3, b, c;
c = f(&b, a);
```

- ( ) 6
- ( ) 2
- ( ) Does not compile
- ( ) 3
- (●) 1

Question 15

1 / 1 pts

Examine this version of the ***swap()*** function. How do you call it?

```
void swap(int * x, int & y)
{
    . . .
}
. . .
int a = 3, b = 7;
// What goes here ?
```

- ( ) `swap(a, &b);`
- ( ) `swap(a, b);`
- ( ) None of these
- ( ) `swap(&a, &b);`
- (●) `swap(&a, b);`