1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr;

cout << \*++ptr;

* 1. 1
  2. **2**
  3. 3
  4. 4

1. What does the expression arr + 2 represent?
   1. The value at the third index of the array
   2. **The address of the third element in the array**
   3. The address of the second element in the array
   4. The value at the second index of the array
2. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr + 3;

cout << \*(ptr - 2);

* 1. 1
  2. **2**
  3. 3
  4. 4

1. Which of the following statements is true about pointer arithmetic?
   1. Pointer arithmetic is not allowed in C++
   2. **Pointer arithmetic operates on the memory addresses of variables**
   3. Pointer arithmetic only works with integer values
   4. Pointer arithmetic requires the use of the -> operator
2. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr;

cout << \*(ptr + 4);

* 1. 1
  2. 2
  3. 3
  4. **5**

1. What is the size of the array int arr[5] in bytes?
   1. 5
   2. 10
   3. 20
   4. **Depends on the size of an int**
2. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = arr;

cout << ptr[1];

* 1. 10
  2. **20**
  3. 30
  4. Compiler error

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr + 1;

cout << \*ptr + 2;

* 1. 2
  2. 3
  3. **4**
  4. 5

1. What does the expression \*(arr + 2) represent?
   1. **The value at the third index of the array**
   2. The address of the third element in the array
   3. The address of the second element in the array
   4. The value at the second index of the array
2. What is the output of the following code snippet?

int arr[5];

cout << sizeof(arr);

* 1. 5
  2. 10
  3. **20**
  4. Depends on the size of an int

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = &arr[4];

cout << ptr[-2];

* 1. 1
  2. **2**
  3. 3
  4. 4

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = &arr[3];

cout << \*(ptr - 3);

* 1. **1**
  2. 2
  3. 3
  4. 4

1. Which of the following correctly declares a pointer to an integer array?
   1. int\* ptr;
   2. int\*[] ptr;
   3. **int (\*ptr)[];**
   4. int\* ptr[];
2. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr;

cout << ptr[3];

* 1. 1
  2. 2
  3. 3
  4. **4**

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = &arr[2];

cout << \*ptr;

* 1. 1
  2. 2
  3. **3**
  4. 4

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr + 2;

cout << ptr[-1];

* 1. 1
  2. **2**
  3. 3
  4. 4

1. Which of the following correctly declares a pointer to a constant integer array?
   1. **const int\* ptr;**
   2. const int\*[] ptr;
   3. const int (\*ptr)[];
   4. const int\* ptr[];
2. What is the output of the following code snippet?

int arr[5];

int\* ptr = arr + 3;

cout << sizeof(ptr);

* 1. **4**
  2. 8
  3. Depends on the size of a pointer
  4. Depends on the size of an int

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = &arr[1];

cout << \*ptr + 1;

* 1. 2
  2. 3
  3. **4**
  4. 5

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = &arr[0];

cout << \*(ptr + 4);

* 1. 1
  2. 2
  3. 3
  4. **5**

1. What is the output of the following code snippet?

int arr[5];

int\* ptr = arr;

cout << sizeof(\*ptr);

* 1. 4
  2. 8
  3. Depends on the size of a pointer
  4. **Depends on the size of an int**

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr;

cout << ptr[2];

* 1. 1
  2. 2
  3. **3**
  4. 4

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr + 4;

cout << \*(ptr - 3);

* 1. 1
  2. **2**
  3. 3
  4. 4

1. Which of the following statements is true about arrays and pointers in C++?
   1. Arrays and pointers are fundamentally the same in C++
   2. Arrays and pointers have the same memory representation
   3. **Arrays can decay into pointers in certain contexts**
   4. Arrays and pointers cannot be used interchangeably
2. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr;

cout << \*(++ptr);

* 1. 1
  2. **2**
  3. 3
  4. 4

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = &arr[3];

cout << \*(ptr + 2);

* 1. **1**
  2. 2
  3. 3
  4. 4

1. What is the output of the following code snippet?

int arr[5];

int\* ptr = &arr[2];

cout << sizeof(ptr);

* 1. **4**
  2. 8
  3. Depends on the size of a pointer
  4. Depends on the size of an int

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr + 1;

cout << \*(++ptr);

* 1. 1
  2. 2
  3. **3**
  4. 4

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = &arr[3];

cout << \*(ptr - 1);

* 1. 1
  2. **2**
  3. 3
  4. 4

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = &arr[2];

cout << \*ptr;

* 1. 1
  2. 2
  3. **3**
  4. 4

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr + 2;

cout << \*(--ptr);

* 1. **1**
  2. 2
  3. 3
  4. 4

1. What is the output of the following code snippet?

int arr[] = {1, 2, 3, 4, 5};

int\* ptr = arr + 3;

cout << \*(ptr + 1);

* 1. 1
  2. 2
  3. 3
  4. **4**

1. Which of the header file is used for array type manipulation?
   1. <array>
   2. <type\_traits>
   3. <iostream>
   4. **std namespace**
2. What is the use of is\_array() function in C++?
   1. **To check if a variable is array type or not**
   2. To check if a variable is 1-D array type or not
   3. To check if a variable is 2-D array type or not
   4. To check if a variable is 1-D or 2-D array type or no
3. The number of values between braces { } can not be larger than the number of elements that we declare for the array between square brackets [ ].
   1. **TRUE**
   2. FALSE
   3. Can be true or false
   4. Can not say
4. What is a pointer?
   1. **A variable that stores the memory address of another variable**
   2. A variable that stores the value of another variable
   3. A variable that stores the size of another variable
   4. A variable that stores the data type of another variable
5. What operator is used to declare a pointer variable?
   1. **\***
   2. &
   3. $
   4. %
6. Which of the following correctly assigns the address of variable "x" to the pointer "ptr"?
   1. ptr = \*x;
   2. ptr = x;
   3. **ptr = &x;**
   4. ptr = address(x);
7. What is the size of a pointer variable in C++?
   1. 1 byte
   2. 2 bytes
   3. 4 bytes
   4. **Depends on the system architecture**
8. What does the "NULL" pointer represent?
   1. **A pointer that points to the address 0**
   2. A pointer that points to the last memory address
   3. A pointer that points to a random memory location
   4. A pointer that doesn't exist
9. What is the result of dereferencing a NULL pointer?
   1. Compilation error
   2. Runtime error
   3. **Undefined behavior**
   4. No effect
10. What is the purpose of dynamic memory allocation in C++?
    1. To allocate memory for global variables
    2. To allocate memory for local variables
    3. To allocate memory for arrays
    4. **To allocate memory at runtime for variables**
11. Which operator is used to allocate memory for a single variable dynamically?
    1. **new**
    2. malloc
    3. alloc
    4. create
12. How do you deallocate memory that has been dynamically allocated using "new"?
    1. **delete ptr;**
    2. dealloc ptr;
    3. destroy ptr;
    4. free(ptr);
13. What is pointer arithmetic?
    1. Performing arithmetic operations on pointers
    2. **Performing arithmetic operations using pointers as operands**
    3. Converting pointers to arithmetic values
    4. None of the above
14. What is the result of incrementing a pointer by 1?
    1. **The pointer moves to the next memory location based on the data type**
    2. The pointer moves to the previous memory location based on the data type
    3. The pointer moves to the next memory location irrespective of the data type
    4. The pointer remains unchanged
15. What is the data type of a pointer that points to an integer?
    1. int
    2. integer
    3. **int\***
    4. integer\*
16. What is the result of subtracting one pointer from another?
    1. **The difference in the memory addresses divided by the size of the data type**
    2. The difference in the memory addresses
    3. The sum of the memory addresses divided by the size of the data type
    4. The sum of the memory addresses
17. What is the size of "int\*" on a 32-bit system?
    1. 2 bytes
    2. **4 bytes**
    3. 8 bytes
    4. Depends on the compiler
18. What is the use of the "sizeof" operator with pointers?
    1. **It returns the size of the data type pointed to by the pointer**
    2. It returns the size of the pointer variable
    3. It returns the size of the memory block allocated to the pointer
    4. It returns the size of the memory address pointed to by the pointer
19. What is the difference between "ptr++" and "++ptr"?
    1. **"ptr++" increments the pointer after accessing the value, while "++ptr" increments the pointer before accessing the value**
    2. "ptr++" increments the pointer before accessing the value, while "++ptr" increments the pointer after accessing the value
    3. Both "ptr++" and "++ptr" increment the pointer after accessing the value
    4. Both "ptr++" and "++ptr" increment the pointer before accessing the value
20. What does the expression "ptr + 1" represent?
    1. **The memory address after the current pointer location**
    2. The memory address before the current pointer location
    3. The value of the variable pointed to by the pointer plus 1
    4. The value of the variable pointed to by the pointer minus 1
21. Which of the following operations is NOT allowed on void pointers?
    1. **Dereferencing**
    2. Arithmetic operations
    3. Casting to other pointer types
    4. None of the above
22. What is the correct way to allocate memory for an array dynamically?
    1. **int\* arr = new int[size];**
    2. int arr[size];
    3. int\* arr = malloc(size \* sizeof(int));
    4. int arr[size] = new int[];
23. What is the purpose of "const" in a pointer declaration?
    1. It makes the pointer constant, meaning it cannot be reassigned to point to another memory location
    2. **It makes the pointed-to value constant, meaning it cannot be modified through the pointer**
    3. It makes the pointer and the pointed-to value both constant
    4. It has no effect on the pointer declaration
24. What is the output of the following code snippet?

int arr[5] = {1, 2, 3, 4, 5};

int\* ptr = arr;

cout << \*(ptr + 3);

* 1. 1
  2. 2
  3. 3
  4. **4**

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = &arr[0];

cout << \*(++ptr);

* 1. 10
  2. **20**
  3. 30
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = arr;

cout << \*ptr++;

* 1. **10**
  2. 20
  3. 30
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = arr;

cout << \*(ptr++) + \*(++ptr);

* 1. 30
  2. 40
  3. **50**
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = arr + 1;

cout << \*(ptr - 1);

* 1. **10**
  2. 20
  3. 30
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = arr + 1;

cout << \*ptr++;

* 1. 10
  2. **20**
  3. 30
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = arr;

cout << \*++ptr;

* 1. 10
  2. **20**
  3. 30
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = arr;

cout << \*ptr + \*ptr++;

* 1. 20
  2. 30
  3. **40**
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = &arr[0];

cout << \*++ptr;

* 1. 10
  2. **20**
  3. 30
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = arr;

cout << \*ptr + \*(ptr + 2);

* 1. 20
  2. 30
  3. **40**
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = &arr[1];

cout << \*ptr--;

* 1. 10
  2. **20**
  3. 30
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = arr;

cout << \*ptr-- + \*ptr;

* 1. 10
  2. 20
  3. **30**
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = &arr[2];

cout << \*(--ptr);

* 1. 10
  2. **20**
  3. 30
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = arr;

cout << \*(ptr + 1) + \*(ptr + 2);

* 1. 20
  2. 30
  3. **40**
  4. Compiler error

1. What is the output of the following code snippet?

int arr[3] = {10, 20, 30};

int\* ptr = &arr[0];

cout << \*(ptr + 1) - \*ptr;

* 1. 1
  2. **10**
  3. 20
  4. Compiler error

1. Pointers are used primarily for:
   1. Declaring variables
   2. **Allocating memory**
   3. Looping structures
   4. Input/output operations