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