Cpp introduction quiz

1) Which of the following is not a primitive data type?

a) char

b) integer

c) string

d) long

2) Which of the following operation not correctly matches with the name of the operator?

a) ‘&&’ → Logical operator

b) ‘!=’ → logical operator

c) ‘+’ → arithmetic operator

d) ‘<=’ → relational operator

3) Which of these is not a legitimate way of decision making in c?

a) if, else, else if

b) The goto statement

c) The ternary operator

d) All are legitimate

4) Which of the function is not a part of the string library?

a) strcmp

b) strlen

c) strncmp

d) strsplt

5) Which of the following statements correctly highlights the difference between arrays and pointers in C?

a) Arrays can be directly reassigned to point to different memory locations, whereas pointers are fixed once initialized.

b) Arrays can be used in arithmetic expressions, such as incrementing or decrementing their values, whereas pointers do not support such operations.

c) Arrays have a fixed size determined at compile-time, whereas pointers can dynamically adjust their size during runtime.

d) Arrays can be passed as function arguments by value, whereas pointers are always passed by reference.

6) Which of the following statements about namespace pollution is true?

A) Namespace pollution occurs when symbols are left undefined in a namespace.
B) Namespace pollution is the process of organizing code into logical groups.
C) Namespace pollution only happens when using nested namespaces.
D) Namespace pollution refers to leaving symbols in a namespace where they shouldn't be.

7) Which prefix is used to represent octal literals in C++?

A) 0x
B) 0b
C) 0o
D) 0d

8) Which segment of memory stores static variables in C++?

A) Code segment
B) Stack segment
C) Heap segment
D) Data segment

9) Which of the following statements about user-defined functions in C++ is NOT true?

A) User-defined functions are written by programmers to perform specific actions.
B) User-defined functions consist of function declarations, function definitions, and function calls.
C) User-defined functions can only be called using the call-by-value mechanism.
D) User-defined functions increase code reusability and make the code more readable.

10) Which of the following statement(s) is/are true regarding function overloading in C++?

A) Function overloading allows multiple functions with the same name but different number of arguments.
B) Function overloading allows multiple functions with the same name but different types of arguments.
C) Function overloading allows multiple functions with the same name but different order of arguments.
D) Function overloading allows multiple functions with the same name but different return types.

Choose the correct option(s):
A) A only
B) B only
C) A and B
D) A, B, and C

11) Which of the following statements is true regarding static polymorphism in C++?

A) Static polymorphism is achieved through function overloading.
B) Static polymorphism is achieved through function overriding.
C) Static polymorphism allows objects of different classes to be treated as objects of the same class.
D) Static polymorphism is determined at runtime.

12) Which of the following statements regarding the string and vector types in C++ is true?

A) The string type is a fixed-length sequence of characters, whereas the vector type is a variable-length sequence of elements.
B) Both the string and vector types are defined as part of the C++ standard library.
C) To use the string type, include the vector header and vice versa.
D) The string type is defined in the std namespace, while the vector type is defined in the std::vector namespace.

13) Which of the following statements regarding the use of the "auto" keyword and the "decltype" type specifier in C++ is true?

A) The "auto" keyword is used to specify the type of a variable at compile time, while the "decltype" type specifier is used to infer the type of a variable at runtime.
B) The "auto" keyword is used to infer the type of a variable from its initializer, while the "decltype" type specifier yields the type of a specified expression.
C) The "auto" keyword is used to declare a function template whose return type depends on the types of its template arguments, while the "decltype" type specifier is used to declare regular functions.
D) The "auto" keyword can be used to infer the type of a variable only if it is initialized, otherwise, a compile-time error occurs. The "decltype" type specifier does not have any such restrictions.

14) Consider the following code snippet in C++:

```
class Shape {
protected:
    double area;
public:
    virtual void calculateArea() = 0;
    double getArea() {
        return area;
    }
};

class Circle : public Shape {
private:
    double radius;
public:
```

```
    Circle(double r) : radius(r) {}
    void calculateArea() override {
        area = 3.14 * radius * radius;
    }
};

class Rectangle : public Shape {
private:
    double length;
    double width;
public:
    Rectangle(double l, double w) : length(l), width(w) {}
    void calculateArea() override {
        area = length * width;
    }
};

int main() {
    Circle circle(5.0);
    Rectangle rectangle(3.0, 4.0);
    Shape* shape1 = &circle;
    Shape* shape2 = &rectangle;
    shape1->calculateArea();
    shape2->calculateArea();
    double totalArea = shape1->getArea() + shape2->getArea();
    return 0;
}
```

In object-oriented programming (OOP) concepts, which principle is demonstrated in the given code snippet?

A) Encapsulation
B) Inheritance
C) Polymorphism
D) Abstraction

15) Which of the following problems can be effectively solved through the use of Garbage Collection in programming languages?

A) Memory fragmentation problem
B) Stack overflow problem
C) Circular reference problem
D) Buffer overflow problem

Choose the correct option:
A) A and B
B) B and C
C) C and D
D) A, C, and D

code snippet based questions

1) Which of the following code snippets correctly defines a variable 'age' of type 'int' in C++ and assigns it the value 27?

a) int age = "27";

b) double age = 27.0;

c) char age = '27';

d) int age = 27;

2) Which of the following code snippets demonstrates a side effect in C programming?

a)

```
int square(int x) {
    return x * x;
}
```

b)

```
int multiply(int a, int b) {
    int result = a * b;
    return result;
}
```

c)

```
int increment(int *x) {
    (*x)++;
    return *x;
}
```

d)

```
void printMessage(char *message) {
    printf("%s\n", message);
}
```

3) Consider the following code snippet in C:

```
const char* message1 = "Hello";
char* const message2 = "World";
```

Which of the following statements accurately describes the properties of the variables `message1` and `message2`?

A) Both `message1` and `message2` are mutable pointers to mutable characters/strings.
B) `message1` is a mutable pointer to an immutable character/string, while `message2` is an immutable pointer to a mutable character/string.
C) `message1` is an immutable pointer to an immutable character/string, while `message2` is a mutable pointer to a mutable character/string.
D) `message1` is a mutable pointer to a mutable character/string, while `message2` is an immutable pointer to an immutable character/string.

4) Which of the following is an example of setting an environment variable in C?

A) `putenv("PATH=/usr/bin")`
B) `getenv("HOME")`

C) `unsetenv("TEMP")`

D) `setenv("LANG", "en_US", 1)`

5) Which of the following options demonstrates the correct way to open a file named "data.txt" in C++ for reading using the input file stream (`ifstream`)?

A) `ifstream input("data.txt", ios::in);`

B) `ifstream input.open("data.txt", ios::in);`

C) `ifstream input.open("data.txt");`

D) `ifstream input("data.txt");`

6) What is the scope of the variable "x" in the following code snippet?

```
int x = 5;
void myFunction() {
    int x = 10;
    // ...
}
int main() {
    // ...
}
```

A) Global scope

B) Local scope within myFunction()

C) Local scope within main()

D) Both global and local scope

7) What will be the output of the following code?

```
#include <iostream>

void myFunction() {
    static int count = 0;
    count++;
```

```
    std::cout << "Count: " << count << std::endl;
}

int main() {
    myFunction();
    myFunction();
    myFunction();

    return 0;
}
```

A) Count: 0
Count: 1
Count: 2

B) Count: 1
Count: 2
Count: 3

C) Count: 3
Count: 3
Count: 3

D) Count: 1
Count: 1
Count: 1

8) Consider the following code snippet:

```
#include <iostream>

int main() {
    auto x = 5;
    auto y = 3.14;
    auto z = 'A';

    std::cout << "x: " << x << std::endl;
    std::cout << "y: " << y << std::endl;
    std::cout << "z: " << z << std::endl;

    return 0;
}
```

What will be the output of the above code?

A) x: 5
y: 3.14
z: A

B) x: 5
y: 3.140000
z: 65

C) x: 5
y: 3.14
z: 65

D) x: 5
y: 3.14
z: A

9) Consider the following code snippet:

```
#include <iostream>

int globalVariable = 10;

void incrementGlobalVariable() {
    globalVariable++;
}

int main() {
    std::cout << "Initial globalVariable: " << globalVariable << std::endl;
    incrementGlobalVariable();
    std::cout << "After increment: " << globalVariable << std::endl;

    return 0;
}
```

What will be the output of the above code?

A) Initial globalVariable: 10
After increment: 11

B) Initial globalVariable: 0
After increment: 1

C) Initial globalVariable: 10
After increment: 10

D) Initial globalVariable: 0
After increment: 10

10) Consider the following code snippet:

```
#include <iostream>

int multiply(int a, int b);

int main() {
    int x = 5;
    int y = 3;
    int result = multiply(x, y);
    std::cout << "Result: " << result << std::endl;

    return 0;
}

int multiply(int a, int b) {
    return a * b;
}
```

What will be the output of the above code?

A) Result: 8
B) Result: 15
C) Result: 3
D) Compilation error

11) Consider the following code snippet:

```
#include <iostream>

void swap1(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}

void swap2(int& a, int& b) {
```

```
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    int x = 10;
    int y = 20;

    swap1(x, y);
    std::cout << "After swap1: x = " << x << ", y = " << y << std::endl;

    swap2(x, y);
    std::cout << "After swap2: x = " << x << ", y = " << y << std::endl;

    return 0;
}
```

What will be the output of the above code?

A) After swap1: x = 10, y = 20
After swap2: x = 20, y = 10

B) After swap1: x = 20, y = 10
After swap2: x = 20, y = 10

C) After swap1: x = 10, y = 20
After swap2: x = 10, y = 20

D) After swap1: x = 20, y = 10
After swap2: x = 10, y = 20

12) Consider the following code snippet:

```
#include <iostream>

void display(int x) {
    std::cout << "Integer value: " << x << std::endl;
}

void display(double x) {
    std::cout << "Double value: " << x << std::endl;
}

void display(char x) {
    std::cout << "Character value: " << x << std::endl;
}
```

```
int main() {
    int a = 10;
    double b = 3.14;
    char c = 'A';

    display(a);
    display(b);
    display(c);

    return 0;
}
```

What will be the output of the above code?

A) Integer value: 10
Double value: 3.14
Character value: A

B) Double value: 10
Double value: 3.14
Character value: A

C) Integer value: 10
Integer value: 3.14
Character value: A

D) Integer value: 10
Double value: 3.14
Double value: A

13) Which of the following function declarations in C++ demonstrates the use of both reference parameters and variable number of arguments?

A) void func(int& arg1, double& arg2, ...);
B) void func(int arg1, double arg2, ...);
C) void func(int& arg1, double arg2, ...);
D) void func(int arg1, double& arg2, ...);

Choose the correct option:
A) A
B) B

C) C
D) D

14) Which of the following code snippets correctly demonstrates the usage of a template function in C++?

A)

```
cppCopy code
template <typename T>
T add(T a, T b) {
    return a + b;
}

int result = add(5, 10);
```

B)

```
cppCopy code
template <typename T>
void print(T value) {
    cout << value << endl;
}

print("Hello, World!");
```

C)

```
cppCopy code
template <class T>
T multiply(T a, T b) {
    return a * b;
}

double result = multiply(3.14, 2.0);
```

D)

```
cppCopy code
template <class T>
void display(T item) {
    cout << item << endl;
}

display(42);
```

15) Consider the following C++ code snippet:

```
cppCopy code
int* func() {
    int* ptr = new int[5];
    return ptr;
}

void foo() {
    int* arr = func();
    delete[] arr;
}

int main() {
    foo();
    // ...
}
```

What is the potential issue with the given code snippet related to dynamic memory management?

A) The `delete[]` operator is missing, causing a memory leak.
B) The `new` operator is missing, resulting in a compilation error.
C) The `delete` operator is used instead of `delete[]`, leading to undefined behavior.
D) The `new[]` operator is used to allocate a single `int`, causing a mismatched deallocation.

16) Consider the following code:

```
cppCopy code
#include <cstdlib>#include <iostream>int main() {
    int* ptr = (int *)malloc(sizeof(int));
    free(ptr);
    ptr = NULL;

    // Some code here

    return 0;
}
```

What is the state of the pointer `ptr` after the code segment `// Some code here`?

A) `ptr` is a valid pointer to the previously allocated memory
B) `ptr` is a dangling pointer
C) `ptr` is set to `NULL`
D) `ptr` is deallocated and cannot be accessed anymore