

Name/Grade/Section: _____

SUMMATIVE TEST

1. Insert the missing part of the code below to output "Hello World!".

```
int main() {  
    _____ << "Hello World!";  
    return 0;  
}
```

2. Insert a new line after "Hello World", by using a special character:

```
int main() {  
    cout << "Hello World! _____";  
    cout << "I am learning C++";  
    return 0;  
}
```

3. Comments in C++ are written with special characters. Insert the missing parts:

```
_____ This is a single-line comment  
_____ This is a multi-line comment _____
```

4. Create a variable named `myNum` and assign the value `50` to it.

```
_____ = _____;
```

5. Display the sum of `5 + 10`, using two variables: `x` and `y`.

```
_____ = _____;  
int y = 10;  
cout << x + y;
```

6. Create a variable called `z`, assign `x + y` to it, and display the result.

```
int x = 5;
int y = 10;
  = x + y;
cout << ;
```

7. Fill in the missing parts to create three variables of the same type, using a **comma-separated list**:

```
 x = 5  y = 6  z = 50;
cout << x + y + z;
```

8. Use the correct keyword to get user input, stored in the variable `x`:

```
int x;
cout << "Type a number: ";
 >> ;
```

9. Fill in the missing parts to print the sum of two numbers (which is put in by the user):

```
int x, y;
int sum;
cout << "Type a number: ";
 >> ;
cout << "Type another number: ";
 >> ;
sum = x + y;
cout << "Sum is: " << ;
```

10. Add the correct data type for the following variables:

```
 myNum = 9;  
 myDoubleNum = 8.99;  
 myLetter = 'A';  
 myBool = false;  
 myText = "Hello World";
```

11. Create two boolean variables, named yay and nay, and add appropriate values to them:

```
  = ;  
  = ;
```

12. Create a greeting variable, and display the value of it:

```
  = "Hello";  
cout << ;
```

13. Multiply 10 with 5 , and print the result.

```
cout << 10  5;
```

14. Divide 10 by 5 , and print the result.

```
cout << 10  5;
```

15. Use the correct operator to increase the value of the variable x by 1 .

```
int x = 10;  
x;
```

16. Use the **addition assignment** operator to add the value 5 to the variable x.

```
int x = 10;  
x  5;
```

17. Fill in the missing part to create a greeting variable of type string and assign it the value Hello.

```
  =  ;
```

18. Use the correct operator to **concatenate** two strings:

```
string firstName = "John ";  
string lastName = "Doe";  
cout << firstName  lastName;
```

19. Use the **correct function** to print the length of the txt string.

```
string txt = "Hello";  
cout << .  ;
```

20. Access the **first character** (H) in myString and print the result:

```
string myString = "Hello";  
cout <<  ;
```

21. Change the **first character** (H) in myString to 'J':

```
string myString = "Hello";  
   ;  
cout << myString;
```

22. Use the **correct function** to read a line of text which is put in by the user.

```
string fullName;  
cout << "Type your full name: ";  
 (cin, fullName);  
cout << "Your name is: " << fullName;
```

23. Use the correct function to print the **highest value** of `x` and `y`.

```
int x = 5;  
int y = 10;  
cout <<  (x, y);
```

24. Use the correct function to print the **square root** of `x`.

```
#include <iostream>  
#include <>  
using namespace std;  
  
int main() {  
    int x = 64;  
    cout <<  (x);  
    return 0;  
}
```

25. Use the correct function to round the number 2.6 to its nearest integer.

```
#include <iostream>
#include <[ ]>
using namespace std;

int main() {
    cout << [ ](2.6);
    return 0;
}
```

26. Fill in the missing parts to print the values 1 (for true) and 0 (for false):

```
[ ] isCodingFun = true;
[ ] isFishTasty = false;
cout << [ ];
cout << [ ];
```

27. Fill in the missing parts to print the value true (for true):

```
int x = 10;
int y = 9;
cout << ([ ] [ ] [ ]);
```

28. Print "Hello World" if x is greater than y .

```
int x = 50;
int y = 10;
[ ] (x [ ] y) {
    cout << "Hello World";
}
```


29. Print "Hello World" if `x` is equal to `y`.

```
int x = 50;
int y = 50;
    (x    y) {
    cout << "Hello World";
}
```

30. Print "Yes" if `x` is equal to `y`, otherwise print "No".

```
int x = 50;
int y = 50;
    (x    y) {
    cout << "Yes";
}    {
    cout << "No";
}
```

31. Print "1" if `x` is equal to `y`, print "2" if `x` is greater than `y`, otherwise print "3".

```
int x = 50;
int y = 50;
    (x    y) {
    cout << "1";
}    (x    y) {
    cout << "2";
}    {
    cout << "3";
}
```

32. Insert the missing parts to complete the following "short hand if...else statement" (**ternary operator**):

```
int time = 20;
string result =    time < 18    "Good day."    "Good evening.";
cout << result;
```