CPP-Conditional Statements Revision

Question 1:

What will be the output of the following code?

```
int x = 5;
if (x = 0)
   cout << "Zero";
else
   cout << "Non-Zero";
```

A) Zero

B) Non-Zero

C) Compilation Error

D) Runtime Error

▼ Solution

> Answer: B) Non-Zero

Explanation: The condition `if (x = 0)` is an assignment, not a comparison. It assigns `0` to `x`, which evaluates to `false`, so the `else` block executes, printing `"Non-Zero"`.

Question 2:

What will be the output of the following code?

```
int x = 10, y = 20;
if (x++ > 10 && y-- > 20)
```

```
    cout << "Inside If";
else
    cout << "Inside Else";

cout << " " << x << " " << y;
```

A) Inside If 11 19

B) Inside If 10 20

C) Inside Else 11 19

D) Inside Else 11 20

▼ Solution

> Answer: D) Inside Else 11 20

Explanation:

- `x++ > 10` evaluates to `false` because `x++` increments after comparison.
- Due to short-circuiting, `y-- > 20` is not evaluated.
- `x` becomes `11`, `y` remains `20`, and `"Inside Else"` is printed.

Question 3:

What will be the output of the following code?

```
int x = 1;
if (x-- && ++x)
    cout << "True";
else
    cout << "False";
```

A) True

B) False

C) Compilation Error

D) Undefined Behavior

▼ Solution

Answer: A) True

Question 4:

What will be the output of the following code?

```
int x = 5;
if (x & 1)
    cout << "Odd";
else
    cout << "Even";
```

A) Odd

B) Even

C) Compilation Error

D) Runtime Error

▼ Solution

Answer: A) Odd

Explanation:

- The bitwise AND operator `&` checks if the least significant bit is `1` (odd) or `0` (even).
- `5 & 1` → `101 & 001` → `001` (true), so `"Odd"` is printed.

Question 5:

What will be the output of the following code?

```
int x = 0, y = 1, z = 2;
if (x || y && z)
    cout << "Yes";
else
    cout << "No";
```

A) Yes

B) No

C) Compilation Error

D) Undefined Behavior

▼ Solution

> Answer: A) Yes

Explanation:

- `y && z` is evaluated first due to precedence (`&&` has higher precedence than `||`).
- Since both `y` and `z` are non-zero, `y && z` evaluates to `true` (`1`).
- `x || 1` is `true`, so `"Yes"` is printed.

Question 6:

What will be the output of the following code?

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

int main() {
    int x = 5, y = 10;
    if (++x > 5 || ++y > 10)
        cout << "Inside If ";
    else
        cout << "Inside Else ";

    cout << x << " " << y;
}
```

A) Inside If 6 10

B) Inside If 6 11

C) Inside Else 6 10

D) Inside Else 6 11

▼ Solution

Answer: A) Inside If 6 10

Explanation:

- `++x > 5` → `6 > 5` → **true**
- `||` short-circuits, so `++y > 10` is **not evaluated**.
- Hence, `y` **remains** `10`.
- `"Inside If"` prints and the final values are `x = 6, y = 10`.

Question 7:

What will be the output of the following code?

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

int main() {
    int a = 3, b = 5, c = 7;
    int result = (a > b) ? (b > c ? b : c) : (a > c ? a : c);
    cout << result;
}
```

A) 3

B) 5

C) 7

D) Compilation Error

▼ Solution

Answer: C) 7

Explanation:

- `(a > b) → (3 > 5) → false`, so it picks the second part: `(a > c ? a : c)`.
- `(3 > 7) → false`, so it picks `c`, which is `7`.

Question 8:

What will be the output of the following code?

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

int main() {
    int x = 0;
    if (x = 5) {
        if (x < 10)
            cout << "X is Small ";
        if (x == 5)
            cout << "X is Five ";
        else
            cout << "X is Something Else ";
    } else {
        cout << "Inside Else ";
    }
}
```

A) X is Small X is Five

B) Inside Else

C) X is Something Else

D) Compilation Error

▼ Solution

Answer: A) X is Small X is Five

Explanation:

- `if (x = 5)` → This is an **assignment** (`=`) not comparison (`==`).
- So, `x` gets **assigned** `5`, which is `true`, so the first `if` executes.
- `if (x < 10)` → `5 < 10` → **true**, so `"X is Small"` prints.
- `if (x == 5)` → **true**, so `"X is Five"` prints.
- `else` belongs to `if (x == 5)`, but it never runs.

Question 9:

What will be the output of the following code?

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

int main() {
    for (int i = 0; i < 7; i++) {
        if (i == 3)
            continue;
        if (i == 5)
            break;
        cout << i;
    }
}
```

A) 01246

B) 0124

C) 01234

D) 01245

▼ Solution

Answer: B) 0124

Explanation:

- `i = 3` → `continue` → skips printing `3`.
- `i = 5` → `break` → loop terminates, so `5` is never printed.
- Output: `0124`.

Question 10:

What will be the output of the following code?

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

```
int main() {
    int a = 4, b = 5, c = 6;
    cout << ((a < b) ? a : b) + c;
}
```

A) 4

B) 9

C) 10

D) 11

▼ Solution

> Answer: C) 10

Explanation:

- `(a < b) ? a : b` → `4 < 5` → **picks** `a` **(4)**.
- `4 + 6 = 10`.

Question 11:

What will be the output of the following code?

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

int main() {
    int x = 2, y = 3;
    if (x++, y++, x + y == 7)
        cout << "Condition True ";
    else
        cout << "Condition False ";

    cout << x << " " << y;
}=
```

A) Condition True 3 4

B) Condition False 3 4

C) Condition True 2 3

D) Compilation Error

▼ Solution

> Answer: A) Condition True 3 4

Explanation:

- `if (x++, y++, x + y == 7)` :
  - `x++ → x = 3`
  - `y++ → y = 4`
  - `x + y == 7` → `3 + 4 == 7` → **true**.
- `"Condition True 3 4"` prints.

Happy Coding!