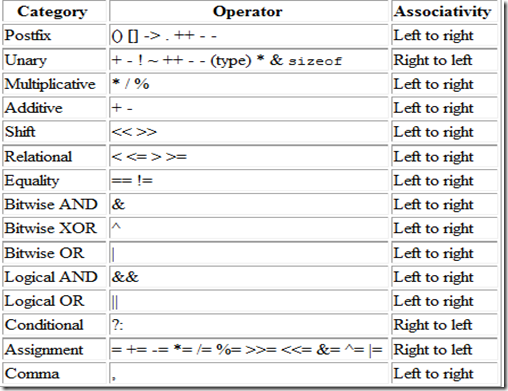
**Bitwise Operators:**

It does bitwise manipulation

| Operator | Description |
| --- | --- |
| **&** | **Bitwise AND Operator** |
| **|** | **Bitwise OR Operator** |
| **^** | **Bitwise XOR Operator** |
| **~** | **Bitwise Complement Operator** |
| **<<** | **Bitwise Shift Left Operator** |
| **>>** | **Bitwise Shift Right Operator** |

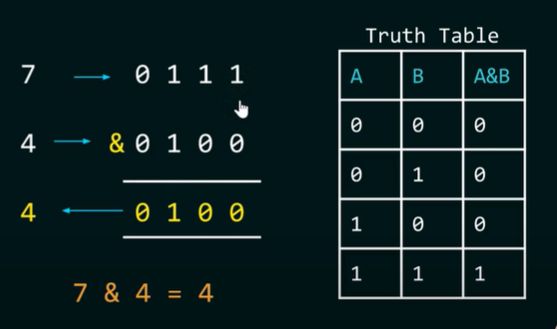
**Note:** Bitwise operators can only be used alongside **char** and **int** data types.



**1) AND Operator ( & ):**

The bitwise AND & operator returns 1 if and only if both the operands are 1. Otherwise, it returns 0. It is a binaryoperator (takes 2 numbers).

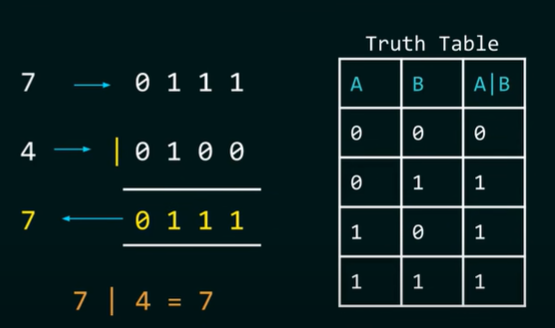
**Example:**



**2) OR Operator ( | ):**

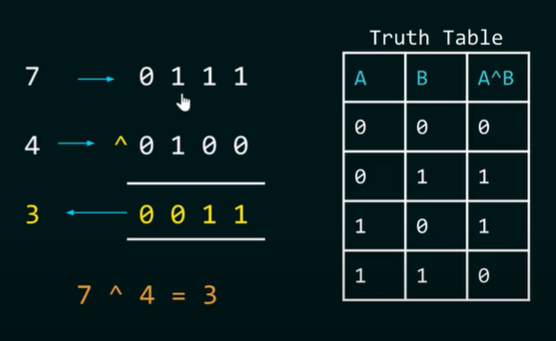
The bitwise OR | operator returns 1 if at least one of the operands is 1. Otherwise, it returns 0.

**Example:**



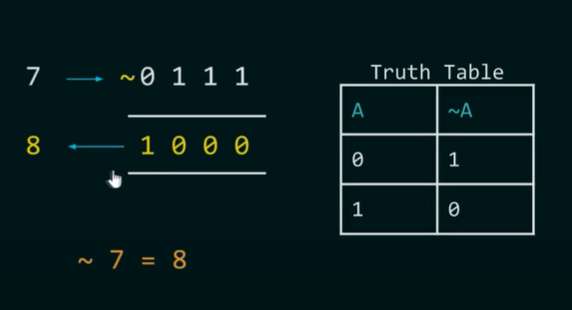
**3) XOR Exclusive Operator ( ^ ):**

The bitwise XOR ^ operator returns 1 if and only if one of the operands is 1. OHowever, if both the operands are 0, or if both are 1, then the result is 0.

****

**4) Complement / NOT Operator ( ~ ):**

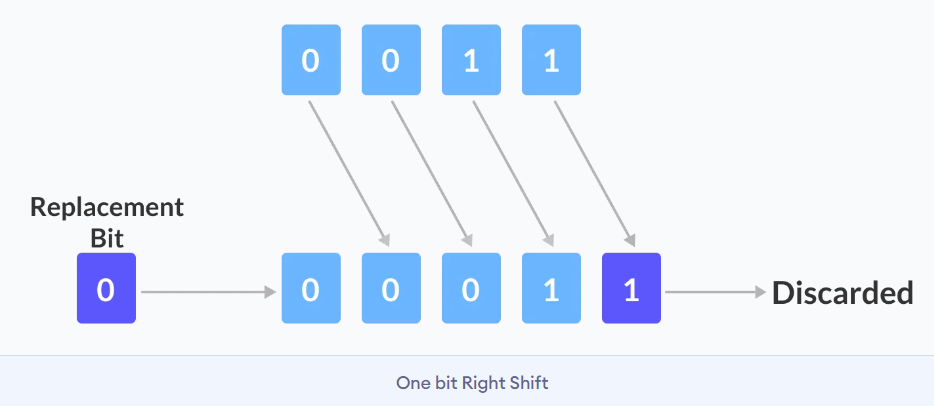
The bitwise complement operator is a **unary** operator (works on only one operand). It is denoted by ~ that changes binary digits 1 to 0 and 0 to 1.



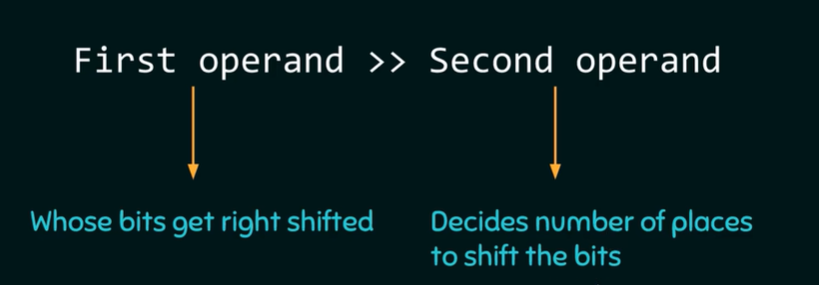
**4) Right Shift Operator ( >> ):**

The right shift operator shifts all bits towards the right by a certain number of specified bits. It is denoted by >>.

When we shift any number to the right, the **least significant bits are discarded**, while the most significant bits are replaced by zeroes.



It requires 2 operands.



**Left shift = division by ( 2^rightOperand)**

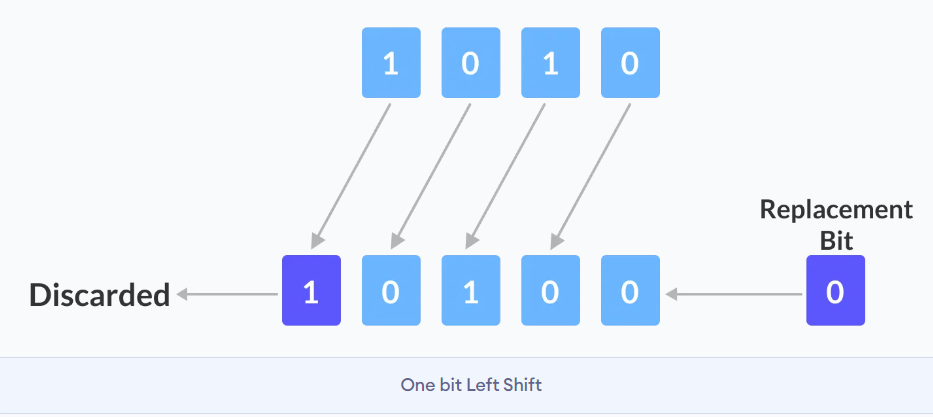
Int num = 3= (0000 0011)

num>>1

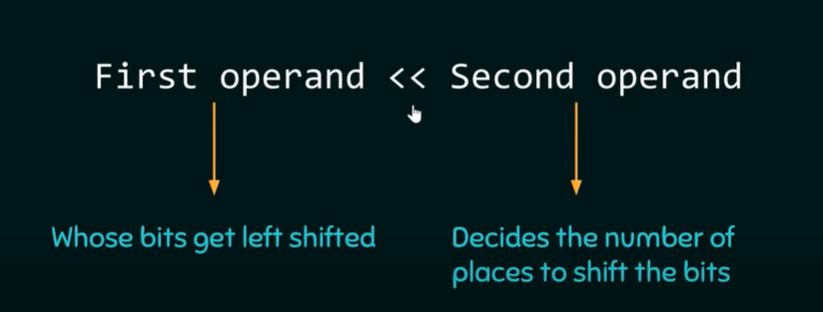
Output = 1 = (3 / 2¹) = (0000 0001)

**4) Left Shift Operator ( << ):**

The left shift operator shifts all bits towards the left by a certain number of specified bits. It is denoted by <<.



It requires 2 operands.



**Left shift = multiplication by ( 2^rightOperand)**

Int num = 3 = (0000 0011)

num<<1

Output = = (3x 2¹) = (0000 0110)

As a result, the bitwise left-shift operation for any number can be different depending on the number of bits they are represented by.

Because in **32-bit representation**, there are many more bits that can be shifted left when compared to **4-bit representation**.

**BIT MASKING**

### **1) Extracting Bits**

To extract specific bits from a number, you can use the & (bitwise AND) operator along with a bitmask. A bitmask is a binary pattern that determines which bits to keep and which to clear.

#include <iostream>

using namespace std;

int main() {

int num = 29; // 0001 1101 in binary

int mask = 4; // 0000 0100 in binary

// Use bitwise AND to extract the 3rd bit (from the right)

int result = num & mask; // Result will be 4 if the bit is set, or 0 if it's not

if (result != 0)

cout << "The 3rd bit is set." << endl;

else

cout << "The 3rd bit is not set." << endl;

return 0;

}

### **2. Setting Bits**

To set specific bits in a number, use the | (bitwise OR) operator with a mask that has the desired bits set to 1.

#include <iostream>

using namespace std;

int main() {

int num = 29; // 0001 1101 in binary

int mask = 2; // 0000 0010 in binary (setting 2nd bit)

// Use bitwise OR to set the 2nd bit

int result = num | mask; // Result will be 31 (0001 1111)

cout << "After setting the 2nd bit: " << result << endl;

return 0;

}

### **3. Clearing Bits**

To clear specific bits (set them to 0), use the & operator with a mask that has 0 in the positions of the bits you want to clear and 1 elsewhere. The mask can be generated using the bitwise NOT operator ~.

**CODE:**

#include <iostream>

using namespace std;

int main() {

int num = 29; // 0001 1101 in binary

int mask = ~(1 << 3); // Create a mask to clear the 4th bit (from the right)

// Use bitwise AND to clear the 4th bit

int result = num & mask; // Result will be 21 (0001 0101)

cout << "After clearing the 4th bit: " << result << endl;

return 0;

}

### **4. Toggling Bits**

To toggle specific bits (flip them between 0 and 1), use the ^ (bitwise XOR) operator with a mask that has 1 in the bit positions to be toggled.

**CODE:**

#include <iostream>

using namespace std;

int main() {

int num = 29; // 0001 1101 in binary

int mask = 8; // 0000 1000 in binary (toggling the 4th bit)

// Use bitwise XOR to toggle the 4th bit

int result = num ^ mask; // Result will be 21 (0001 0101)

cout << "After toggling the 4th bit: " << result << endl;

return 0;

}

### **Summary of Bitwise Operators:**

* **&** (AND): Extracts bits.
* **|** (OR): Sets bits.
* **^** (XOR): Toggles bits.
* **~** (NOT): Inverts bits.

**PRACTICE QUESTIONS**

**Question 1:**

By Areen Zainab

//code to swap 2 number using bitwise operators

#include <iostream>

using namespace std;

int main() {

int a = 10; //1010

Int b = 15; //1111

a = a ^ b; // Step 1

b = a ^ b; // Step 2

a = a ^ b; // Step 3

cout << " a = " << a << " , b = " << b;

return 0;

}

**What will be the output of this question?**

1. a= 10 , b= 15
2. a= 15, b=10
3. a= 5, b= 10
4. a=10, b=5

**Answer: b**

**Question:**

**Bit masking:**

#include <iostream>

using namespace std;

int main() {

int num = 29; // 0001 1101 in binary

int mask = 2; // 0000 0010 in binary (setting 2nd bit)

// Use bitwise OR to set the 2nd bit

int result = num | mask; // Result will be 31 (0001 1111)

cout << "After setting the 2nd bit: " << result << endl;

return 0;

}

**//precedence questions**

**Question 2:**

By Areen Zainab

#include <iostream>

using namespace std;

int main() {

int a = 5; // 5 in binary: 0101

int b = 3; // 3 in binary: 0011

int result = a ^ b & a;

cout << result;

return 0;

}

**What will be the output of the following code?**

A) 0

B) 1

C) 4

D) 5

**Answer:** C

**Question 3:**

By Areen Zainab

#include <iostream>

using namespace std;

int main() {

int x = 12; // 12 in binary: 1100

int y = 10; // 10 in binary: 1010

int result = x | y ^ y;

cout << result;

return 0;

}

**What will be the output of the following code?**

A) 12

B) 10

C) 14

D) 6

**Answer:** A

**Question 4;**

By Afnan Rizwan

#include <iostream>

using namespace std;

int main() {

int a = 18; //10010

int b = 7; //00111

int result = (a ^ b) & (~(a & b));

cout << "Result = " << result << endl;

return 0;

}

**Output:**

Result = 21

**Question 5;**

By Afnan Rizwan

#include <iostream>

using namespace std;

int main() {

int num = 14; //

int shiftValue = 2;

int result = (num << shiftValue) | (num >> (4 - shiftValue));

cout << "Final Result: " << result << endl;

return 0;

}

**Output:**

Final Result: 59

**Question 6;**

By Areen Zainab

Write a C++ function to count the number of set bits (1s) in the binary representation of an integer using bitwise operators.

#include <iostream>

using namespace std;

int main() {

int num = 29; // 29 in binary: 11101

int count = 0;

while (num > 0) {

count += num & 1; // Check if the least significant bit is 1

num = num >> 1; // Shift the number to the right by 1

}

cout << "Number of set bits: " << count;

return 0;

}

**What will be the output of this question?**

**Answer:** Number of set bits: 4

**Question 7:**

By Areen Zainab

#include <iostream>

using namespace std;

int main() {

int a = 6; // 6 in binary: 0110

int result = a ^ a >> 1;

cout << result;

return 0;

}

**What will be the output of the following code?**

A) 2

B) 4

C) 5

D) 3

**Answer:** C

**Question 8:**By Rimsha Azam

Create a C++ program that implements a bitwise algorithm to find the maximum element in an array without using comparison operators.

Sample Output:

Input: {10, 25, 5, 30}

Output: "The maximum element is 30."

**Question 9:**By Zubair Adnan

int main() {

unsigned short int num = 5; // for your ease. short int is 16 bits

int move = 6 ; //0110

switch(move) {

case 1 ... 16:

num = num ^ (1 << move-1);

break;

default:

num = num & move;

break;

}

cout << “Number = “ << num;

return 0;

}

**Solution:** 37

**Question 10:**

By Zubair Adnan

#include <iostream>

using namespace std;

int main() {

unsigned short int num = 10;

switch(num & 1) {

case 0:

if(num ^ ~(num & num-1))

Cout << “Space”;

else

cout << “Marines”;

break;

case 1:

cout << "Terra";

default:

cout << "Sanguini";

break;

}

return 0;

}

**Solution:** Space

**Question 11:**

By Daniyal Aziz

Clear the last right side set a bit of a number

uint8\_t remove\_last\_set\_bit( uint8\_t number )

{

return (n & (n - 1));

}

**Question 12:**

By Daniyal Aziz

**Swap all even and odd bits**

unsigned int swapOddEvenBits( unsigned int number )

{

return ( ( number & 0xAAAAAAAA ) >> 1 ) | ( ( number & 0x55555555 ) << 1 );

}

**Question 13:**

By Aneeq Malik

Write the output of the following programs as:

Output-1: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Output-2: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Output-3: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Output-4: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include <iostream>

#include <bitset>

using namespace std;

int main()

{

unsigned short int n = 55534;

// binary conversion of 55534 bits is

// 1101100011101110

// Note that bitset<4> (5) means 0101

int size = sizeof(short int) \* 8;

int bits = 9;

int mask = (1 << bits) - 1;

mask <<= (size - bits);

cout << bitset<sizeof(short int) \* 8>(mask) << endl; // output-1

mask = (n & mask) >> (size - bits);

cout << bitset<sizeof(short int) \* 8>(mask) << endl; // output-2

n <<= bits;

cout << bitset<sizeof(short int) \* 8>(n) << endl; // output-3

n |= mask;

cout << bitset<sizeof(short int) \* 8>(mask) << endl; // output-4

}

**Answer:**

1111111110000000

0000000110110001

1101110000000000

0000000110110001

**Question 14:**

By Aneeq Malik

Write the output of the following programs as:

Output-1: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Output-2: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Output-3: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Output-4: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**.**

**.**

**.**

**.**

#include <iostream>

#include <bitset>

using namespace std;

int main()

{

unsigned short int n = 52345;

// binary conversion of 52345 bits is

// 1100110011011001

int size = sizeof(unsigned short int) \* 8;

int bits1 = 6, bits2 = 4, bits3 = 3;

// First Mask (Extract bits1)

int mask1 = (1 << bits1) - 1;

mask1 <<= (size - bits1);

cout << bitset<sizeof(unsigned short int) \* 8>(mask1) << endl; // output-1

// Extract bits1 from n and shift right

unsigned short int extracted1 = (n & mask1) >> (size - bits1);

cout << bitset<sizeof(unsigned short int) \* 8>(extracted1) << endl; // output-2

// Second Mask (Extract bits2)

int shiftBits = (size - bits1 - bits2);

int mask2 = (1 << bits2) - 1;

mask2 <<= shiftBits;

cout << bitset<sizeof(unsigned short int) \* 8>(mask2) << endl; // output-3

// Extract bits2 from n and shift right

unsigned short int extracted2 = (n & mask2) >> shiftBits;

cout << bitset<sizeof(unsigned short int) \* 8>(extracted2) << endl; // output-4

// Third Mask (Extract bits3)

int mask3 = (1 << bits3) - 1;

cout << bitset<sizeof(unsigned short int) \* 8>(mask3) << endl; // output-5

// Extract bits3

unsigned short int extracted3 = (n & mask3);

cout << bitset<sizeof(unsigned short int) \* 8>(extracted3) << endl; // output-6

n <<= 3;

cout << bitset<sizeof(unsigned short int) \* 8>(n) << endl; // output-7

n |= extracted1;

cout << bitset<sizeof(unsigned short int) \* 8>(n) << endl; // output-8

n ^= extracted3;

cout << bitset<sizeof(unsigned short int) \* 8>(n) << endl; // output-9

return 0;

}

**Output:**

1111110000000000

0000000000110011

0000001111000000

0000000000000001

0000000000000111

0000000000000001

0110001111001000

0110001111111011

0110001111111010

**Question 15:**

By Aneeq Malik

Write a program that swaps all odd and even bits in a given 16-bit number. For example, swap bit 0 with bit 1, bit 2 with bit 3, and so on.

***Solution:***

#include <iostream>

#include <bitset>

using namespace std;

int main()

{

unsigned short int n = 43690; // 1010101010101010 in binary

cout << "Before Swapping:\n" << bitset<16>(n) << endl;

// Mask for even bits: 0101010101010101 decimal = 21845

// Mask for odd bits: 1010101010101010 decimal = 43690

// you can confirm the above with calculator

unsigned short int even\_bits = n & 21845;

unsigned short int odd\_bits = n & 43690;

// Shift even bits left and odd bits right

even\_bits <<= 1;

odd\_bits >>= 1;

// Combine even and odd bits

unsigned short int result = even\_bits | odd\_bits;

cout << "After Swapping:\n" << bitset<16>(result) << endl;

return 0;

}

**Output:**

Before Swapping:

1010101010101010

After Swapping:

0101010101010101

**Question 16:**

By Aneeq Malik

**Write a program that checks if a given number is a power of 2.**

#include <iostream>

using namespace std;

int main() {

unsigned short int n = 1024; // 0000010000000000 in binary

if ((n != 0) && ((n & (n - 1)) == 0))

cout << "It is a power of 2." << endl;

else

cout << "It is NOT a power of 2." << endl;

return 0;

}

**Switch Case Statements:**

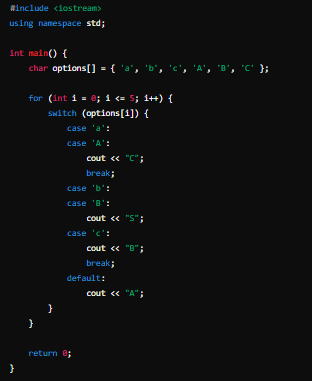
1. **Switch Case Questions  
     
   Question 1:**By Rimsha Azam

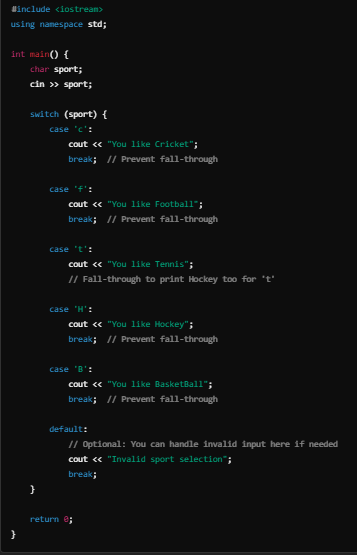
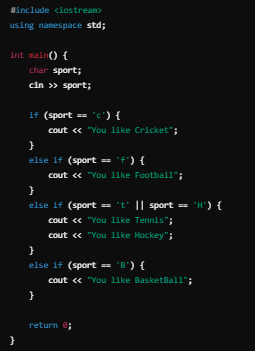
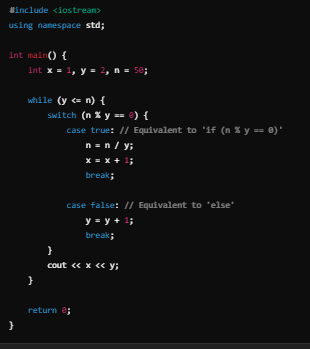
Write a C++ program that prompts the user to enter a positive integer. Use a switch case statement to determine if the number is a prime number, a perfect square, or neither. Additionally, the program should handle cases where the input is negative or zero.

Sample Output:

Input: 25

Output: "The number 25 is a perfect square."  
  
**Question 2:**By Rimsha Azam

  
Output : CSBBCSBA

**Question 3:**By Rimsha Azam  
  
Convert the following code to switch cases.  
   
  
CODE ^^ SOLUTION^^  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Question 4:**By Rimsha Azam  
  
  
  
Output : 222324253545

**Question 1:**By Muhammad Shayan

What will be the output of the following code?

int x = 8;

switch (x) {

case 1: cout << "One ";

case 2: cout << "Two ";

case 8: cout << "Eight ";

case 10: cout << "Ten ";

default: cout << "Default";

}

A) Eight Ten Default

B) Default

C) Eight Default

D) No output

Answer : A

**Question 2:**By Muhammad Shayan

#include <iostream>

using namespace std;

int main() {

int value = 1;

switch (value) {

case 1:

cout << "One" << endl;

case 2:

cout << "Two" << endl;

break;

case 3:

cout << "Three" << endl;

break;

default:

cout << "Default" << endl;

}

return 0;

}

A) One  
B) Two  
C) One Two  
D) Default

Answer: C)

**Question 2:**By Muhammad Shayan

int main() {

int x = 5; // Binary: 0101

switch (x << 1) {

case 2: cout << "Two" << endl; break;

case 5: cout << "Five" << endl; break;

case 10: cout << "Ten" << endl; break;

default: cout << "None" << endl;

}

A) Two

B) Five

C) Ten

D) None

Answer: C) Ten

**Question 1;**

By Areen Zainab

#include <iostream>

using namespace std;

int main() {

int num = 3;

switch (num + 2) {

case 1:

cout << "One ";

break;

case 3:

cout << "Three ";

break;

case 5:

cout << "Five ";

default:

cout << "Default";

}

return 0;

}

**What will be the output of the following code?**

A) Five

B) Five Default

C) Default

D) Three Default

**Answer: B**

**Question 2:**

By Areen Zainab

#include <iostream>

using namespace std;

int main() {

int num = 2;

switch (num) {

case 1:

case 2:

case 3:

cout << "Number is small ";

break;

case 4:

case 5:

cout << "Number is medium ";

break;

default:

cout << "Number is large";

}

return 0;

}

**What will be the output of the following code?**

A) Number is small

B) Number is medium

C) Number is large

D) No output

**Answer: A**

**Switch Case Questions:**

By M. Zubair Adnan

1. **Question 01**

Code:

int main() {

int data1=10, data2=11, data3=12;

switch(int a=(data1 + data2 + data3) / 2) {

case 17:

cout << "17" << endl;

break;

case 16:

cout << "16" << endl;

break;

default:

cout << "default" << endl;

}

switch((data1 + data2 + data3) / 2) {

default:

cout << "default" << endl;

break;

case 17:

cout << "17" << endl;

break;

case 16:

cout << "16" << endl;

break;

}

return 0;

}

Solution:

16

16

1. **Question 02**

Code:

int main() {

int a=7,b=6,c=0;

switch(0) {

case 1:

a=6;

b=8;

cout<<a<<endl;

case -10 … 0:

b=a+c;

cout<<b<<endl;

default:

c=b+3;

cout<<c<<endl;

}

cout<<a<<" "<<b<<" "<<c;

}

Solution:

7

10

7 7 10

**Question no. 01:**

By Daniyal Aziz

Check Whether a Character is Uppercase or Lowercase

**Solution:**

| #include <iostream> using namespace std;   int main() {  char ch;    // Input the character  cout << "Enter a character: ";  cin >> ch;    // Check if the character is uppercase or lowercase using a switch statement  switch (ch) {  case 'A' ... 'Z': // Range for uppercase letters  cout << "The character '" << ch << "' is an uppercase letter." << endl;  break;  case 'a' ... 'z': // Range for lowercase letters  cout << "The character '" << ch << "' is a lowercase letter." << endl;  break;  default:  cout << "The character '" << ch << "' is not an alphabetic letter." << endl;  break;  }    return 0; } |
| --- |

**Question no.02:**

What is the error in the following code?

#include <iostream>

using namespace std;

int main() {

double number;

cout << "Enter a number between 1 and 5: ";

cin >> number;

switch (number) {

case 1.0:

cout << "Number is 1" << endl;

break;

case 2.0:

cout << "Number is 2" << endl;

break;

case 3.0:

cout << "Number is 3" << endl;

break;

default:

cout << "Invalid number!" << endl;

}

return 0;

}

Solution:

Switch only works with integral data types like int and char

***Bitwise Operators***

***Switch Statements***

*By: Aneeq Malik*

**Question-1**

Write the output of the following program

#include <iostream>

using namespace std;

int main() {

int x = 2, y = 4, z = 1;

switch (x + y - z) {

case 2:

cout << 1;

break;

case 5:

switch (y \* z - x) {

case 1:

cout << 2;

break;

case 2:

cout << 3;

case 4:

z += x;

cout << 4;

break;

}

x -= z;

switch (x + y) {

case 1:

cout << 5;

case 2:

cout << 6;

case 3:

cout << 7;

break;

}

cout << 8;

break;

default:

cout << 9;

break;

}

return 0;

}

**Question-2**

Write the output of the following program

#include <iostream>

using namespace std;

int main() {

int x = 4, y = 2, z = 3;

switch (x \* y + z) {

case 9:

cout << "A";

break;

case 10:

cout << "B";

break;

case 11:

cout << "C";

switch (x - y) {

break;

case 0 + 3:

cout << "D";

break;

case 1:

cout << "E";

break;

case (3 && 2 \* 0):

cout << "F";

break;

case 2:

cout << "G";

break;

break;

default:

cout << "H";

break;

}

case 12:

cout << "I";

break;

default:

cout << "J";

break;

}

return 0;

}

***Switch Statements***

*By: Afnan Rizwan*

**Question-1**

#include <iostream>

using namespace std;

int main() {

int x = 8, y = 9, z = 10;

switch (int result = (x \* z - y) % 4) {

case 0:

cout << "Zero" << endl;

break;

case 1:

cout << "One" << endl;

break;

case 2:

cout << "Two" << endl;

break;

default:

x=0;

y=7;

z=10;

cout << "Default" << endl;

switch (((x + y \* z) / 10)+ 5) {

case 7:

cout << "Case 7" << endl;

break;

case 12:

cout << "Case 12" << endl;

default:

cout << "Case Default" << endl;

break;

case 11:

cout << "Case 11" << endl;

break;

}

break;

}

return 0;

}

**Output:**

Default

Case 12

Case Default

**Question-2**

#include <iostream>

using namespace std;

int main() {

int alpha = 3, beta = 5, gamma = 0;

switch (gamma) {

case 1:

alpha = 8;

beta = 10;

cout << alpha << endl;

case -5 ... 0:

beta = alpha + gamma;

cout << beta << endl;

default:

gamma = beta - 2;

cout << gamma << endl;

}

cout << alpha << " " << beta << " " << gamma;

return 0;

}

**Output:**

3

1

3 3 1