Bitwise Operators

13

Question *13.1*

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    int i = 32 , j = 0x20, k, l, m ;
    k = i | j ;
    l = i & j ;
    m = k ^ l ;
    printf ( "%d %d %d %d %d\n", i, j, k, l, m ) ;
    return 0 ;
}
```

A. 32 32 32 32 0
B. 0 0 0 0 0
C. 0 32 32 32 32
D. 32 32 32 32 32

Answer

A

Question *13.2*

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    int i = 4, j = 8 ;
    printf ( "%d %d %d\n", i | j & j | i, i | j && j | i, i ^ j ) ;
    return 0 ;
}
```

A. 4 8 0
B. 1 2 1
C. 12 1 12
D. 0 0 0

Answer

C

Question *13.3*

If an *unsigned int* is 4 bytes wide then which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    unsigned int m = 32 ;
    printf ( "%x\n", ~m ) ;
    return 0 ;
}
```

A. ffffffff
B. 00000000
C. fffffdf
D. ddfdddfd

Answer

C

Question *13.4*

If an *unsigned short int* is 2 bytes wide then which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    unsigned short int a = 0xffff ;
    ~a ;
    printf ( "%x\n", a ) ;
    return 0 ;
}
```

A. ffff
B. 0000
C. 00ff
D. ff00

Answer

A

Question *13.5*

Point out the error, if any, in the following program.

```
#include <stdio.h>
int main( )
{
    unsigned int a, b, c, d, e, f ;
    a = b = c = d = e = f = 32 ;
    a <<= 2 ;
    b >>= 2 ;
    c ^= 2 ;
    d |= 2 ;
    e &= 2 ;
    f ~= 2 ;
    printf ( "%x %x %x %x %x %x\n", a, b, c, d, e, f ) ;
    return 0 ;
}
```


Answer

Error is in *f ~= 2*, since there is no operator like ~=.

Question *13.6*

To which numbering system can the binary number 1011011111000101 be easily converted to?

Answer

Hexadecimal, since each 4-digit binary represents one hexadecimal digit.

Question *13.7*

Which bitwise operator is suitable for checking whether a particular bit is on or off?

Answer

The & operator

Question *13.8*

Which bitwise operator is suitable for turning off a particular bit in a number?

Answer

The & operator

Question *13.9*

Which bitwise operator is suitable for putting on a particular bit in a number?

Answer

The | operator

Question *13.10*

On left shifting, the bits from the left are rotated and brought to the right and accommodated where there is empty space on the right? [True/False]

Answer

False

Question *13.11*

Left shifting a number by 1 is always equivalent to multiplying it by 2. [Yes/No]

Answer

No

Question *13.12*

Left shifting an *unsigned int* or *char* by 1 is always equivalent to multiplying it be 2. [Yes/No]

Answer

Yes

Question *13.13*

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    printf ( "%d >> %d %d >> %d\n", 4 >> 1, 8 >> 1 ) ;
    return 0 ;
}
```

A. 4 1 8 1
B. 4 >> 1 8 >> 1
C. 2 >> 4 Garbage Value >> Garbage Value
D. 2 4

Answer

C

Question *13.14*

Assuming a 2-byte integer which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    printf ( "%x\n", -1 << 3 ) ;
    return 0 ;
}
```

A. ffff
B. fff8
C. 0

D. No output

Answer

B

Question *13.15*

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    unsigned int res ;
    res = ( 64 >> ( 2 + 1 - 2 ) ) & ( ~ ( 1 << 2 ) ) ;
    printf ( "%d\n", res ) ;
    return 0 ;
}
```

A. 32
B. 64
C. 0
D. 128

Answer

A

Question *13.16*

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
```



```
{
    printf ( "%d %d\n", 32 << 1, 32 << 0 ) ;
    printf ( "%d %d\n", 32 << -1, 32 << -0 ) ;
    printf ( "%d %d\n", 32 >> 1, 32 >> 0 ) ;
    printf ( "%d %d\n", 32 >> -1, 32 >> -0 ) ;
    return 0 ;
}
```

A. Garbage values
B. 64 32
   0 32
   16 32
   0 32
C. All zeros
D. 8 0
   0 0
   32 0
   0 16

Answer

B

Question *13.17*

Which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    unsigned char i = 0x80 ;
    printf ( "%d\n", i << 1 ) ;
    return 0 ;
}
```

A. 0
B. 256
C. 100
D. None of the above

Answer

B

Question *13.18*

Which of the following statements are correct about the program given below?

```
#include <stdio.h>
int main( )
{
    unsigned int m[ ] = { 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 } ;
    unsigned char n, i ;
    scanf ( "%d", &n ) ;
    for ( i = 0 ; i <= 7 ; i++ )
    {
        if ( n & m[i] )
            printf ( "yes\n" ) ;
    }
    printf ( "\n" ) ;
    return 0 ;
}
```

A. It will put off all bits that are on in the number *n*.
B. It will test whether the individual bits of *n* are on or off.
C. It will put on all the bits that are off in the number *n*.
D. It will report compilation errors in the *if* statement.

Answer

B

Question *13.19*

Which of the following statements are correct about the program given below?

```
#include <stdio.h>
char * fun ( unsigned int num, int base ) ;
int main( )
{
    char *s ;
    s = fun ( 128, 2 ) ;
    s = fun ( 128, 16 ) ;
    printf ( "%s\n", s ) ;
    return 0 ;
}
char * fun ( unsigned int num, int base )
{
    static char buff[33] ;
    char *ptr ;
    ptr = &buff [ sizeof ( buff ) - 1 ] ;
    *ptr = '\0' ;
    do
    {
        *--ptr = "0123456789abcdef"[ num % base ] ;
        num /= base ;
    } while ( num != 0 ) ;
    return ptr ;
}
```

A. It converts a number to a given base.
B. It converts a number to its equivalent binary.
C. It converts a number to its equivalent hexadecimal.
D. It converts a number to its equivalent octal.

Answer

A

Question *13.20*

```
#define CHARSIZE 8
#define MASK(y) ( 1 << y % CHARSIZE )
#define BITSLOT(y) ( y / CHARSIZE )
#define SET( x, y) ( x[BITSLOT(y)] |= MASK(y) )
#define TEST(x, y) ( x[BITSLOT(y)] & MASK(y) )
#define NUMSLOTS(n) ((n + CHARSIZE - 1) / CHARSIZE)
```

Given the above macros how will you

- declare an array *arr* of 50 bits
- put the 20th bit on
- test whether the 40th bit is on or off

Answer

```
char arr[NUMSLOTS(50)] ;
SET(arr, 20 ) ;
if ( TEST ( arr, 40 ) )
```

Question *13.21*

Consider the macros in Question 13.20 above. On similar lines how will you define a macro that will clear a given bit in a bit array?

Answer

```
#define CLEAR(x, y) ( x[BITSLOT(y)] &= ~MASK(y) )
```


Question *13.22*

Which of the following is the correct output for the program given below?

```
#define P printf ( "%d\n", ~1^~0 ) ;
#define M(P) int main( )\
            {\
                P\
                return 0 ; \
            }
M ( P )
```

A. 1
B. 0
C. -1
D. 2

Answer

A

Question *13.23*

Which of the following statements are correct about the program given below?

```
#include <stdio.h>
int main( )
{
    unsigned int num ;
    int c = 0 ;
    printf ( "Enter a number: " ) ;
    scanf ( "%u", &num ) ;
    for ( ; num ; num >>= 1 )
    {
```

```
        if ( num & 1 )
            c++ ;
    }
    printf ( "%d\n", c ) ;
    return 0 ;
}
```

A. It counts the number of bits that are on in the number *num*.
B. It sets all bits in the number *num* to 1.
C. It sets all bits in the number *num* to 0.
D. None of the above

Answer

A

Question *13.24*

Assuming a 2-byte integer which of the following is the correct output for the program given below?

```
#include <stdio.h>
int main( )
{
    printf ( "%x\n", -1 >> 4 ) ;
    return 0 ;
}
```

A. ffff
B. 0fff
C. 0000
D. fff0

Answer

A. On computers that don't support sign extension, you may get B.

Question *13.25*

In the statement *expression1* >> *expression2* if *expression1* is a signed integer with its leftmost bit set to 1 then on right-shifting it the result of the statement will vary from computer to computer. [True/False]

Answer

True

Question *13.26*

Which of the following statements are correct about the program given below?

```
#include <stdio.h>
int main( )
{
    unsigned int num ;
    int i ;
    scanf ( "%u", &num ) ;
    for ( i = 0 ; i < 16 ; i++ )
        printf ( "%d", ( num << i & 1 << 15 ) ? 1 : 0 ) ;
    printf ( "\n" ) ;
    return 0 ;
}
```

A. It prints all even bits from *num*.
B. It prints all odd bits from *num*.
C. It prints binary equivalent of *num*.
D. None of the above.

Answer

C

Question 13.27

Write a program that rotates the number by *n* number of bits? For example, if binary equivalent of a number is 101011 then after shifting rightmost 3 bits to the left it should become 011101?

Answer

```
#include <stdio.h>
int main( )
{
    int num, n, count = 0, temp ;
    printf ( "Enter number: " ) ;
    scanf ( "%d", &num ) ;
    printf ( "Enter number of bits: " ) ;
    scanf ( "%d", &n ) ;
    temp = num ;
    while ( temp != 0 )
    {
        temp = temp >> 1 ;
        /* Count total number of bits in number */
        count = count + 1 ;
    }
    /* Obtain number of bits to be rotated */
    temp = ( num >> 00 & ( ~ ( ~0 << n ) ) ) ;
    num = num >> n ;
    temp = temp << ( count - n ) ;
    num = num | temp ;
    printf ( "%d\n", num) ;
    return 0 ;
}
```

Question 13.28

Write a program to put on alternate bits on of a number starting from leftmost bit which is on. For example, if binary equivalent of

a number is 10010100 then after bitwise operations it should change to 10101010.

Answer

```
#include<stdio.h>
#include<conio.h>
int main( )
{
    int num, c = 0 ;
    int a = 1, b ;
    printf ( "Enter the number:\n" ) ;
    scanf ( "%d", &num ) ;

    while ( num != 0 )
    {
        num = num >> 1 ;
        c = c + 1 ;
    }
    b = c - 1 ;
    num = a << b ;
    c = b - 2 ;
    while ( c > 0 )
    {
        num = num | ( a << c ) ;
        c = c - 2 ;
    }
    printf ( "%d\n", num ) ;
    return 0 ;
}
```

Question 13.29

Write a program to invert *n* bits of a number from position *p* without disturbing other bits?

Answer

```
#include <stdio.h>
int main( )
{
    int num, n, count = 0, temp, pos ;
    printf ( "Enter number: " ) ;
    scanf ( "%d", &num ) ;
    printf ( "Enter number of bits and position: " ) ;
    scanf ( "%d %d", &n, &pos ) ;
    while ( n > 0 )
    {
        /* if bit is zero*/
        if ( ( ( num >> pos + 1 - n ) & 01 ) == 0 )
            num  |= 01 <<  pos + 1 - n ;  /* Set it */
        else
        {
            /* if bit is one */
            temp = ~num ;
            temp  |= 01 <<  pos + 1 - n ;   /* set bit in temp */
            /* xoring it will give zero */
            /* at required position */
            temp = num ^ temp ;
            num = num & temp ;
        }
        n-- ;
    }
    printf ( "%d\n", num ) ;
    return 0 ;
}
```

Question *13.30*

What will be the output of the following program?

```
#include <stdio.h>
```



```
int main( )
{
    char c = 48 ;
    int i, mask = 01 ;
    for ( i = 1 ; i <= 5 ; i++ )
    {
        printf ( "%c", c | mask ) ;
        mask = mask << 1 ;
    }
    printf ( "\n" ) ;
    return 0 ;
}
```

Answer

12480

Question *13.31*

State True or False:

A. Bitwise & and | are unary operators.
B. Bitwise & operator can be used to check if a bit in number is set or not.
C. Bitwise & operator can be used to check if more than one bit in a number is on.
D. Bitwise & operator can be used to divide a number by powers of 2.
E. Bitwise & operator can be used in conjunction with ~ operator to turn off 1 or more bits in a number.
F. Bitwise | operator can be used to set a bit in number.
G. Bitwise | operator can be used to set multiple bits in a number.
H. Bitwise | operator can be used to multiply a number by powers of 2.
I. Bitwise operators can be used to perform addition and subtraction.
J. Bitwise operators can be used to generate a random number.

K. Bitwise operators can be used to reverse sign of a number.

Answer

A. True
B. True
C. True
D. False
E. True
F. True
G. True
H. False
I. False
J. False
K. False