Floating Point Issues

4

Question 4.1

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

```
#include <stdio.h>
int main( )
{
    float f = 43.20 ;
    printf ( "%e ", f ) ;
    printf ( "%f ", f ) ;
    printf ( "%g \n", f ) ;
    return 0 ;
}
```

A. 4.320000e+001 43.200001 43.2
B. 4.3 43.22 43.21
C. 4.3e 43.20f 43.00
D. Error

Answer

A

Question 4.2

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

```
#include <stdio.h>
int main( )
{
    double d = 2.25 ;
    printf ( "%e ", d ) ;
    printf ( "%f ", d ) ;
    printf ( "%g ", d ) ;
```

```
    printf ( "%lf \n", d ) ;
    return 0 ;
}
```

A. 2.2 2.50 2.50 2.5
B. 2.2e 2.25f 2.00 2.25
C. 2.250000e+000 2.250000 2.25 2.250000
D. Error

Answer

C

Question 4.3

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

```
#include <stdio.h>
int main( )
{
    float a = 0.7 ;
    if ( a < 0.7 )
        printf ( "C\n" ) ;
    else
        printf ( "C++\n" ) ;
    return 0 ;
}
```

A. The program will output C
B. The program will output C++
C. Compiler will report an error saying a *float* cannot be compared with a *double*
D. Output will be C for 16-bit compilers and C++ for 32-bit compilers

Answer

A

Question *4.4*

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

```
#include <stdio.h>
int main( )
{
    float a = 0.7 ;
    if ( a < 0.7f )
        printf ( "C\n" ) ;
    else
        printf ( "C++\n" ) ;
    return 0 ;
}
```

A. C
B. C++
C. Compiler reports an error saying a *float* cannot be compared with a *double*.
D. Output will be C for 16-bit compilers and C++ for 32-bit compilers

Answer

B

Question *4.5*

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

```
#include <stdio.h>
#include <math.h>
int main( )
{
    printf ( "%f\n", sqrt ( 36.0 ) ) ;
    return 0 ;
}
```

A. 6.0
B. 6
C. 6.000000
D. Error: Prototype of *sqrt( )* not found

Answer

C

Question *4.6*

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

```
#include <stdio.h>
int main( )
{
    float fval = 7.29 ;
    printf ( "%d\n" , ( int ) fval ) ;
    return 0 ;
}
```

A. 0
B. 0.0
C. 7.0
D. 7

Answer

D

Question 4.7

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

```
#include <stdio.h>
int main( )
{
    float *p ;
    printf ( "%d \n", sizeof ( p ) ) ;
    return 0 ;
}
```

A. 2 in 16-bit compiler like TC/TC++, 4 in 32-bit compiler like Visual Studio or gcc
B. 4 in 16-bit compiler like TC/TC++, 2 in 32-bit compiler like Visual Studio or gcc
C. 4 in 16-bit compiler like TC/TC++, 4 in 32-bit compiler like Visual Studio or gcc
D. 2 in 16-bit compiler like TC/TC++, 2 in 32-bit compiler like Visual Studio or gcc

Answer

A

Question 4.8

Which statement will you add in the following program to make it work correctly?

```
#include <stdio.h>
int main( )
{
    printf ( "%f\n", log ( 36.0 ) ) ;
    return 0 ;
}
```

Answer

```
#include <math.h>
```

Question 4.9

Will the following *printf( )*s print the same values for any value of *a*? [Yes/No]

```
#include <stdio.h>
int main( )
{
    float a ;
    scanf ( "%f", &a ) ;
    printf ( "%f\n", a + a + a ) ;
    printf ( "%f\n", 3 * a ) ;
    return 0 ;
}
```

Answer

Yes

Question 4.10

We want to round off *x*, a *float*, to an *int* value. The correct way to do so will be

A. `y = ( int ) ( x + 0.5 ) ;`

B. y = int (x + 0.5) ;
C. y = (int) x + 0.5 ;
D. y = (int) ((int) x + 0.5)

Answer

A

Question *4.11*

Which error are you likely to get when you run the following program in TC/TC++?

```
#include <stdio.h>
int main( )
{
    struct emp
    {
        char name[20] ;
        float sal ;
    } ;
    struct emp e[10] ;
    int i ;
    for ( i = 0 ; i <= 9 ; i++ )
    {
        printf ( "Enter name and salary: " ) ;
        scanf ( "%s %f", e[ i ].name, &e[ i ].sal ) ;
    }
    return 0 ;
}
```

A. Suspicious pointer conversion
B. Floating point formats not linked
C. Cannot use *scanf( )* for structures
D. Strings cannot be nested inside structures

Answer

B

Question *4.12*

What causes the error in problem 4.11 above to occur and how will you rectify the error in the above program?

Answer

When the compiler encounters a reference to the address of a *float*, it sets a flag to have the linker link in the floating point emulator. A floating point emulator is used to manipulate floating point numbers in runtime library functions like *scanf( )* and *atof( )*. There are some cases in which the reference to the *float* is a bit obscure and the compiler does not detect the need for the emulator.

These situations usually occur during the initial stages of program development. Normally, once the program is fully developed, the emulator will be used in such a fashion that the compiler can accurately determine when to link in the emulator.

To force linking of the floating point emulator into an application, just include the following function in your program:

```
void linkfloat ( void )
{
    float a = 0, *b = &a ; /* cause emulator to be linked */
    a = *b ; /* suppress warning - variable not used */
}
```

There is no need to call this function from your program.

Note that this problem does not occur when you build and run the same program using gcc or Visual Studio compiler.

Question 4.13

Which of the following statement correctly obtains the remainder on dividing 5.5 by 1.3?

A. rem = 5.5 % 1.3 ;
B. rem = modf (5.5, 1.3) ;
C. rem = fmod (5.5, 1.3) ;
D. We cannot determine the remainder while doing floating point division.

Answer

C

Question 4.14

Which of the following is valid range of *long double*?

A. 3.4E- 4932 to 1.1E+4932
B. 3.4E- 4932 to 3.4E+4932
C. 1.1E- 4932 to 1.1E+4932
D. 1.7E- 308 to 1.7E+308

Answer

D

Question 4.15

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

```
#include <stdio.h>
#include <math.h>
int main( )
{
    float n = 1.54 ;
    printf ( "%f %f\n", ceil ( n ), floor ( n ) ) ;
    return 0 ;
}
```

A. 2.000000 1.000000
B. 1.500000 1.500000
C. 1.550000 2.000000
D. 1.000000 2.000000

Answer

A

Question 4.16

Which are the three different types of real data types available in C, what are their sizes and what format specifiers are used for them?

Answer

Type	Size	Format Specifier
float	4 bytes	%f
double	8 bytes	%lf
long double	8 bytes	%Lf

These sizes are implementation dependent and are likely to vary from one compiler to another. What is guaranteed is size of *float* would not be bigger than *double* and size of *double* will not be bigger than *long* doubile.

Question 4.17

Which of the following is the correct datatype for the variable *a* in the statement given below?

```
a = 23.45 ;
```

A. *float*
B. *double*
C. *long double*
D. Depends upon the memory model that you are using

Answer

A

Question 4.18

What will you do to treat the constant 3.14 as a *float*?

Answer

Use *3.14f*

Question 4.19

What will you do to treat the constant 3.14 as a *long double*?

Answer

Use *3.14L* or *3.14l*

Question 4.20

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

```
#include <stdio.h>
int main( )
{
    printf ( "%d %d %d\n", sizeof ( 3.14f ), sizeof ( 3.14 ), sizeof ( 3.14l ) ) ;
    return 0 ;
}
```

A. 4 4 4
B. 4 8 8
C. 4 8 10
D. 4 8 12

Answer

B

Also refer 4.16.

Question 4.21

The binary equivalent of 5.375 is

A. 101.101110111
B. 101.011
C. 101011
D. None of the above

Answer

B

Question 4.22

How *floats* are stored in binary form?

Answer

Floating-point numbers are represented in IEEE format. The IEEE format for floating point storage uses a sign bit, a mantissa and an exponent for representing the power of 2. The sign bit denotes the sign of the number—a 0 represents a positive value and a 1 denotes a negative value. The mantissa is represented in binary after converting it to its normalised form. The normalised form results in a mantissa whose most significant digit is always 1. The IEEE format takes advantage of this by not storing this bit at all. The exponent is an integer stored in unsigned binary format after adding a positive integer bias. This ensures that the stored exponent is always positive. The value of the bias is 127 for *float*s and 1023 for *doubles*.

Question *4.23*

A *float* occupies 4 bytes. If the hexadecimal equivalent of these 4 bytes are A, B, C and D, then when this *float* is stored in memory in which of the following order do these bytes get stored?

A. ABCD
B. DCBA
C. 0xABCD
D. Depends on whether the machine has a big-endian or a little-endian architecture

Answer

D

In big-endian architecture the low byte is stored earlier than the high byte, whereas, in little-endian architecture the high byte is stored earlier than the low byte. Thus, in big-endian architecture the float is stored as DCBA, whereas, in little-endian architecture it is stored as ABCD.

Question *4.24*

If the binary equivalent of 5.375 in normalised form is 0100 0000 1010 1100 0000 0000 0000 0000, what will be the output of the following program?

```
#include <stdio.h>
int main( )
{
    float a = 5.375 ;
    char *p ;
    int i ;
    p = ( char * ) &a ;
    for ( i = 0 ; i <= 3 ; i++ )
        printf ( "%02X \n", ( unsigned char ) p[ i ] ) ;
    return 0 ;
}
```

A. 40 AC 00 00
B. 04 CA 00 00
C. 00 00 AC 40
D. 00 00 CA 04

Answer

C

Question *4.25*

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

```
#include <stdio.h>
int main( )
{
    float a = 5.375 ;
    printf ( "%f %e %E \n", a, a, a ) ;
    return 0 ;
}
```

A. 5.375 5.375 5.375
B. 5.375000 5.375000 5.375000
C. 5.375000 5.375000e+000 5.375000E+000
D. 5.375000 5.375000E+000 5.375000e+000

Answer

C