C Formatted Output Answers

Sayak Haldar
IIEST, Shibpur

1) Answer is d) 10 2 5

Explanation: First, the function argument of printf (at line 5) which is another printf will be evaluated. So, the inner printf(which is argument to outer printf) will first print 10 2. Now, the inner printf prints 5 characters all together. So, outer printf will print 5 characters.

2) Answer is c) 10 3 some garbage value

Note: Compiler will throw a warning for line 5 telling %d expects an argument of type int

3) Answer is c) 10 3

Note: Compiler will throw a warning for line 5 saying that too many arguments

4)Answer is b) myworld(note: spaces to the left of myworld)
(i.e. 2 spaces then myworld)

i=9;

printf(“%*s”,i,s) : Here, we can use asterisk (*) to pass the width specifier/precision to printf() dynamically , rather than hard coding it into the format string like:

printf(“%9s”,s);

So, basically i=9;printf(“%*s”,i,s) is a way of doing printf(“%9s”,s) dynamically

Now, that means we fix the width of the string to be printed as at least 9 characters.
So, if the string s is of length< 9 characters, a certain number of white spaces will be added to the left of the string to make the total number of printed character 9.

Here, 2 spaces will be added to the left side of “myworld”

5) Answer is b) Print empty spaces if the string state is less than 10 characters

This statement printf(“%10s”, state); means that we fix the width of the string to be printed as at least 10 characters. So, spaces will be printed to the left side of the string if the length of the string is less than 10 characters
So, correct option is b)

6) Answer is b) It keeps the record of the types of arguments that will follow

7) Answer is d) myw(note:7 spaces before myworld)

```
char *s = "myworld";
int i = 3;
printf("%10.*s", i, s);
```

Now, we fix the width of the final string to be printed as 10 characters.

(so, if the string length is <10 characters, some spaces will be added to the left side of the string to make the total character printed as 10.
.*s: we fix the number of characters to be taken from the string s to print as 3 (we pass the number of characters to be printed as 3 dynamically

8) Answer is b) %e always formats in the format [-]m.dddddd or [-]m.dddddE[+|-]xx where no.of ds are optional and output formatting depends on the argument.

9) Answer is b) /

10) Answer is b)sprintf writes the formatted data into a string.

11) Answer is d) %%

The answer of the 12th question is:

7 (Result of printf("%d\n",b);)

7(two spaces then 7) (Result of printf("%3d\n",b);)
Since, we fix the width of the string to be printed to at least 3 characters. So, if 3 characters are not present to be printed, spaces will be added to the left to the string to be printed

007 (Result of printf("%03d\n",b);)
Here, we fix the width of the string to be printed to at least 3 characters, and also mention if there are <3 characters to be printed 0's will be added to the left

5.10 (%3.2f: we fix the width/length of the total number to 3 digits with 2 digits after decimal point.)

The answer of the 13th question is:

%d (print as a decimal integer)

%6d: (print as a decimal integer with a width of at least 6 wide. So, if there are <6 characters to be printed, spaces will be added to the left to make the total count of the characters printed as 6)

%f: print as a floating point

%4f: print as a floating point with width of at least 4 wide. So, if there are <4 characters to be printed, spaces will be added to the left to make the total count of the characters printed as 4)

%.4f (print as a floating point with a precision of four characters after the decimal point)

%3.2f: print as a floating point with at least 3 wide and a precision of 2 digits (or, 2 digits after the decimal point)

The answer of the 14th question is:

The color: blue (As a result of printf("The color: %s\n", "blue");)

First number: 12345 (As a result of printf("First number: %d\n", 12345);)

Second number: 0025(printf("Second number: %04d\n", 25);)
Third number: 1234 (As a result of printf("Third number: %i\n", 1234);)
Note: %d and %i both format specifier are for integer

Float Number: 3.14(As a result of printf("Float number: %3.2f\n", 3.14159);)
Note: %3.2f :=this fixes the number of total characters to be printed as 3 with 2 digits as precision (after decimal point). Also, note that , no round off takes place in such cases.

Hexadecimal: ff (As a result of printf("Hexadecimal: %x\n", 255);)
Note that: 255 will be converted to its equivalent hexadecimal representation because of %x

Octal: 377 (As a result of printf("Octal: %o\n", 255);)

Unsigned value: 150 (As a result of printf("Unsigned value: %u\n", 150);)

Just print the percentage sign % (As a result of printf("Just print the percentage sign %%\n", 10);)

The answer of the 15th question is:

:Hello, world!:
: Hello, world!:
:Hello, wor:
:Hello, world!:
:Hello, world! :
:Hello, world!:
: Hello, wor:
:Hello, wor :

• The printf(“:%s:\n”, “Hello, world!”); statement prints the string (nothing special happens.)
• The printf(“:%15s:\n”, “Hello, world!”); statement prints the string, but print 15 characters. If the string is smaller the “empty” positions will be filled with “whitespace.”
• The printf(“:%.10s:\n”, “Hello, world!”); statement prints the string, but print only 10 characters of the string.
• The printf(“:%-10s:\n”, “Hello, world!”); statement prints the string, but prints at least 10 characters. If the string is smaller “whitespace” is added at the end. (See next example.)
• The printf(“:%-15s:\n”, “Hello, world!”); statement prints the string, but prints at least 15 characters. The string in this case is shorter than the defined 15 character, thus “whitespace” is added at the end (defined by the minus sign.)
• The printf(“:%.15s:\n”, “Hello, world!”); statement prints the string, but print only 15 characters of the string. In this case the string is shorter than 15, thus the whole string is printed.
• The printf(“:%15.10s:\n”, “Hello, world!”); statement prints the string, but print 15 characters.
If the string is smaller the “empty” positions will be filled with “whitespace.” But it will

only print a maximum of 10 characters, thus only part of new string (old string plus the whitespace positions) is printed.
• The printf(“:%-15.10s:\n”, “Hello, world!”); statement prints the string, but it does the exact same thing as the previous statement, accept the “whitespace” is added at the end.

References:

1) http://www.sanfoundry.com/c-interview-questions-answers/
2) http://stackoverflow.com