Suppose a, b and c are integer variables that have been assigned the values **a = 9, b = 34** and **c = -8.** Determine the value of each of the following arithmetic expressions.

Review a c operator precedence table before attempting the next 11 questions

1) b* (c % b)

   Answer: -272

   Explanation

   Begin by evaluating the brackets (c % b) replacing variables with values we get (-8 % 34).
   This is simply the remainder when -8 is divided by 34.
   Finding remainders with –ve signs may be confusing.
   But you can easily get the remainder by using some math.
   -8/34 = 0; (integer division… see quest 2 for details)
   -8 = 34*0 – 8
   So the remainder is -8.
   Now we evaluate b*(c % b). Replacing variables with values we get 34*(-8) which is -272.

2) a/c

   Answer: -1

   Explanation

   Integer division always leads to truncation of the fractional part of the result. So instead of -1.125, the fractional part .125 is truncated and we have -1.

3) a % c

   Answer: 1

   Explanation

   The expression is 9%-8 and we know 9/-8 = -1 (see question 2) so 9 = -8*-1 +1 hence the remainder is

4) a / b * c

   Answer: 0

   Explanation:

   We have no parentheses and both / and * have the same level of precedence so we resort to associativity rules. Since they are both left to right associative, we just evaluate the expression from left to right.

a/b will be 9/34 but as we have seen in question 2, this will give 0 (integer division). Now 0*-8 will still be 0. Hence the answer.

A C program contains the following declarations and initial assignments:

```
int i= 1, j = 4, k; float x = 0.007, y = -0.03, z;
char c = ' e ' , d = 'b ';
```

Determine the value of each of the following expressions. Use the values initially assigned to the variables for each expression.

5) (3 * j - 2 * i) % (2 * d + c)

Answer: 10;

Explanation:

The first bracket is easy to evaluate we first do the multiplications and then the subtraction last. This is the same as (3*4)-(2*1) and yields 10. For the second bracket 2 is multiplied by a character d (d holds a character) so an integral promotion is done on d to implicitly cast it to an integer (The ASCII value for 'b' which is 62). The same is done for the addition with c. The reader should not that the conversion is compiler dependent but the value from the second bracket will be greater than that of the first bracket (10).
So the answer is 10.

6) 3 * ((i / 7) + (4 * (j - 2)) % (i + j - 4))

Answer: 0

Explanation:

3 * ((i / 7) + (4 * (j - 2)) % (i + j - 4))
Replacing variables with values
3 * ((1 / 7) + (4 * (4 - 2)) % (1 + 4 - 4))
Evaluating innermost brackets
3 * (0+ (4 * 2) % (1))
Evaluating % first due to higher priority
3 * (0+ 0)

And 3*0 is 0

7) (x <= y) && (i < 0) || (j < 5)

Answer: 1 //true

Explanation

(x <= y) is false and (i < 0) is false too but (j < 5) is true so we have :

 0 && 0 || 1

Which is 1.

8) z += (x >= 0) ? x : 0

Answer: 0.007

Explanation:
Even though we have not explicitly initiated z, the compiler will provide a default initialization of 0 for us. Since x>=0, 0.007 is added to z

9) (x == y) && (i > 0) && (j != 5)

Answer: 0

Explanation:
Since x and y are not equal, we just short circuit the evaluation.

10) k = (j ==5) ? i : j

Answer: 4

Explanation:
Since j is not equal to five, the value of j is stored in k.

11) i /= (j > 0) ? j : 0

Answer: 0

Explanation:

Since j is greater than zero the conditional operator evaluates to the value of j. hence we have i/j and from integer division of 1/4 we get 0.

A C program contains the following variable declarations. **float a = 9.5, b = 0.0003, c = 6100.**; Show the output resulting from each of the following printf statements. Use underscores (_) to represent every intentionally left space.

12) printf("%f %f %f", a, b, c);

```
Ans: 9.500000_0.000300_6100.000000
```

An underscore represents an actual blank space

```
13) printf('%3f %3f %3f", a, b, c);
          Ans: 9.500000_0.000300_6100.000000

14) printf ("%8f %8f %8f", a, b, c);
      Ans: 9.500000_0.000300_6100.000000

15) printf("%l2.4e %12.4e %12.4e", a, b, c);
      Ans: 2.4e__9.5000e+000_3.0000e-004

16)printf( "%g %g %g", a, b, c);
      Ans: 9.5_0.0003_6100

17)scanf("%9d %9d %9d",&i &j &k);

18)scanf("%8d %8o %8x",&i &j &k);

19)scanf("%6o %6o %6x",&i &j &k);

20)printf("%s",text);

21)printf("%.4s",text);

22)printf("%*.4s",8,text);
```

23. ANS

```
1    4     9    16   25
X = 25
```

HINT: This program is requesting you to glean from 0 to 9 as values of 'i' (i < 10) and check whether it is an even number
"if (i % 2==0)". If the number is an even number, you add this value to the current value of 'x' and add 1 to the sum and display it. The results is stored in 'x' and used for the rest of the values of 'i'. The last value of 'x' should be displaced on a new line as shown.

24. ANS

```
2,1    5,3   10,5   17,7   26,9
X = 26
```

HINT: This program is similar to that of Q. 23, just that over here, we are interested in the odd numbers i.e
"if (i % 2==0)". If the test is positive, you add the value

of ‘i’ to ‘x’ (and stored in ‘x’ as well) and then display the value of ‘i’ and ‘x’ as shown above. The last value of ‘x’ is displaced on a new line as usual.

25. ANS

```
5,5
X=5
```

HINT: ‘i’ should be divisible by 5 i.e. “if (i % 5 == 0)”. Gleaning is from 1 to 9 since i is initialized as “i=1”. The process is then similar to Q.24

26. ANS

```
1,1    0,2   3,3   2,4
X = 2
```

HINT: This program is requesting you to check whether the value of ‘i’ is odd i.e.
“if (i%2==1)” before you add its value to ‘x’. If the value of ‘i’ is not odd, then you decrease the value of ‘x’ by 1. In both cases you are tasked to display the value of ‘x’ alongside the value of ‘i’ as shown. After executing the program for ‘i’ = 4 i.e “(i< 5)”, you display the value of ‘x’ on a new line as shown.

27. ANS

```
1,1    0,2   3,3   2,4
X = 2
```

HINT: Same as Q. 26. The *continue* added to this program literally means “do not stop”.

28. ANS

```
1,1
X = 1
```

HINT: Same as Q. 26. The *break* added to this program literally means “stop after first cycle”

29. ANS

```
1,1    2,2   3,4   4,8
X = 4
```

HINT: This program expects you to glean some integer values for 'i' in such a way that i<10; 'i' is initialize at 1. (meaning 1 is the first value for 'i'). You go ahead and test whether this value is less than 10 as requested. If it is, you increase the value of 'x' by 1 (x++). From here, you do the display of the two values as we have seen in other questions. You will then note that instead of our usual i++ in the "for function", it is rather i*=2, (This means you multiply the current value of 'i' by two and then store it as the current value of 'i'). You go on with the process by making sure 'i' is not greater than 10. Then you display the last value of 'x' on a new line.

30. ANS

```
-1    -1     0    -1      -1
```

HINT: This program make use of functions. We are required to count from 1 to 5
(count<= 5). With each count, the function is called. We now go to where the function is to perform an operation with the count's number. We are then asked to display the results of the operation done in the function.

31. ANS

```
1,1    2,2     3,3
```

HINT: Same explanation as Q. 30. But remember that we do not roll over on the value of 'y'.

32. ANS

```
10, 16      -10, 4
```

HINT: Same explanation as Q. 30. We are doing the counting from 1 to 2 (count<3). Two global variables 'a' and 'b' has been declared and assigned the values 10 and 20 respectively. There are also two local variables, c and d which depends on the count for its value. In the printf, we encounter the use of a function called funct1. We then go to where the function is and perform a required operation with the given variables; ‘a’,’b’,’c’ and‘d’. It is worth to note that whenever a function is called, you stop momentarily to go and perform the necessary operation required in the function.

33. ANS

```
2  4  8  12  18  24  24  32  32  32
```

HINT: Over here, we will be employing arrays. An array ‘c’ has been declared as containing some integers. You are required to take notice of the positions of each integer in the array ‘c’. The first integer is having the 0th position and the last integer is having the 9th position. Two variables, ‘a’ and ‘b’ has also been declared as integers. What the program does is that it count up ‘a’ from 0 to 9 i.e. “a<10”. At each count it goes and look for the corresponding position from the array c, and then pick up that value. If the value is an even number i.e. “if c[a] % 2 ==0”, it goes ahead and add this value to the current value of ‘b’ and then display it. Where the value is odd, the old value of ‘b’ is displayed.

34. ANS

```
1  2  5  8  13  18  25  25  34   34
```

HINT: Same approach like the previous. Note that this program deals with the odd integers in the array though. i.e. “if c[a] % 2 ==1”. So where the value is even, the old value of ‘b’ is maintained.

35. ANS

```
1  1  4  4   9
```

HINT: Same approach like the Q.33

36. ANS

```
1
```

HINT: Explanation is given on a sheet of paper.

37. ANS

```
12  11  10  9  8  7  6  5  4  3  2  1
```

HINT: Explanation is given on a sheet of paper.