1. **What is output of following program?**

**main()**

**{**

**int x = 1;**

**int y = 2;**

**switch(x | y)**

**{**

**case 1: x = 2;**

**case 2: x = 3;**

**case 3: x = 4;**

**case 4: x = 5;**

**default:x = 6;**

**}**

**printf("%d\n", x);**

**}**

**Choices:**

**a)6**

**b)4**

**c)5**

**d)3**

a. There are no break statements so, irrespective of the value of x, last case (default) will be executed. This is an example of fall-through situation.

2. What is the output of the following program?

main()

{

printf("%d\n", 100 / 10 / 10);

}

 a)1

b)100

c)10

d)0

a. For most of operators in C, the order of evaluation is from left to right. (exceptions: unary, ternary and assignment operators). So, 100/10/10 is evaluated as (100/10)/10 which is equal to 1.

3. main()

{

int a=5;

if (a=1)

{

printf("%d", a);

}

}

In the above code

a)The printf statement will never get executed

b)The printf statement will always get executed and the value of a will be printed as 5

c)The program will encounter syntax error

d)The printf statement will always get executed and the value of a will be printed as 1

* Tricky!! D is the answer. The printf statement will always get executed and the value of a will be printed as 1

4. int main()

{

int a[10];

int i;

for (i=0; i<10; i++)

a[i] = i;

printf("%d", a[-1]);

}

The above program

a)Will encounter a compilation error

b)Will encounter segmentation violation when run

c)Will have unpredictable behavior

d)Will get into infinite loop

* the correct answer is c. The array a[10] is in stack. So, a[-1] will point to some location within stack and its content is unpredictable.

5. What is the output of the following program?

main()

{

int a=3, b=5, c=1;

int x=8;

x = a < (b < c);

printf("%d", x);

}

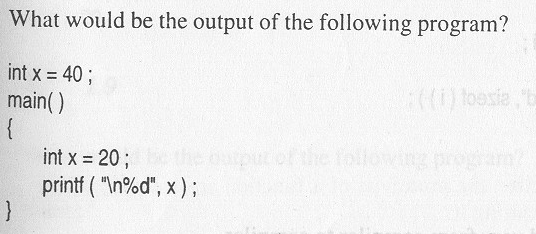
a)1

b)8

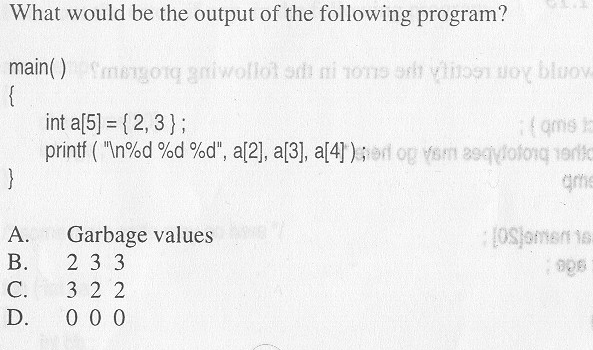
c)Will encounter run time error

d)0

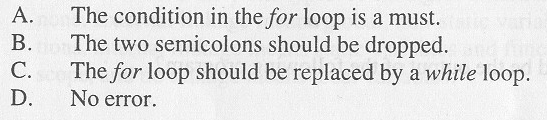
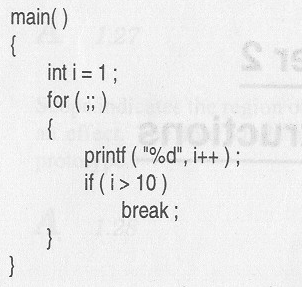
* d. The expression (b < c) is evaluated first. It is equivalent to (5 < 1) and this expression will return 0. So, the next part of expression will become a < 0 which is equivalent to 3 < 0 and hence result is 0. This result is assigned to x and hence value of x becomes 0.



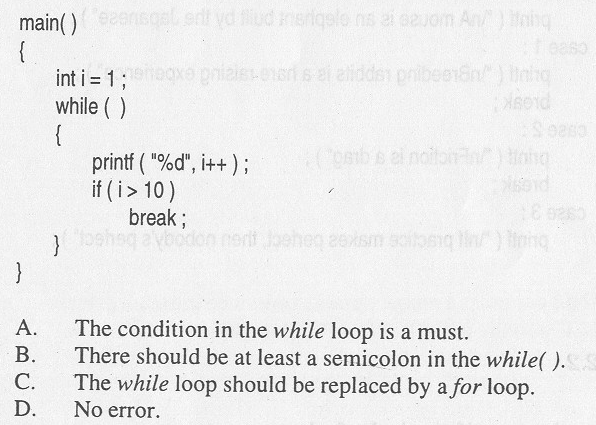
* 20 is the answer.. Scope matters!



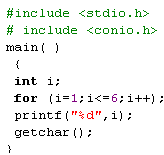
When partial initialization is done, you are all set to 0 ☺



D is the answer.. And the output is 12345678910



A is the answer





for (i=1;i<=10;i++); is similar to

for (i=1;i<=10;i++)

{

}

10. main()

{

int i=7;

switch(i)

{

case 1:

printf("one");

break;

case 2\*3+1:

printf("seven");

break;

}

}

Answer:

Seven

11. #include <stdio.h>

main( )

{

int x=10, y=20, z=5,i;

i=x<y<z;

printf ("\n %d", i);

getchar();

}

Answer:

(10<20)<5

So answer is 1

12. main()

{

int arr[]={1,2,3,4,5};

printf("%d",sizeof(arr));

}

Answer:

20

13. What is the output of the following program?

int x[100];

main()

{

printf("%d\n",x[99]);

}

 a)Unpredicatable

b)Runtime error

c)0

d)99

The correct answer is c. In C, it is guaranteed that all the uninitialized global variables are initialized to 0.

14. How many times is "Hello world" is printed in following program.

main()

{

unsigned int i = 5;

while (--i >= 0)

{

printf("Hello World\n");

}

}

 a)5

b)6

c)Infinite

d)Program will not compile

* The correct answer is c. Being an unsigned number, value of i cannot be less than 0. So, the statement i >= 0 will be always true. Hence, this program goes into an infinite loop.

15. What is the output of following program?

main()

{

int y = 100;

const int x = y;

printf("%d\n", x);

}

a)100

b)Garbage value

c)Error

d)0

* The correct answer is a. It is ok to initialise a const variable with another variable. The keyword const indicates that once initialised, it shouldn't be changed.

16.int main()

{

int j=6;

int i;

i = (5, j);

printf("%d %d\n", i, j);

return 0;

}

The output from the above program is:

a)5 6

b)6 6

c)6 7

d)The output is machine dependent

* The correct answer is b. (5, j) is first evaluated as (5, 6). When placed in parenthesis (5,6) would return 6 which gets assigned to i.

17.. int main()

{

int j=6;

int i;

i = (5, j++);

printf("%d %d\n", i, j);

return 0;

}

The output from the above program is:

a)5 6

b)6 6

c)6 7

d)The output is machine dependent

* The correct answer is c. (5, j++) is first evaluated as (5, 6). When placed in parenthesis (5,6) would return 6 which gets assigned to i. Then j is incremented.

18.What is the output of following program?

main()

{

int x = 0;

int y = 0;

if (x++ && ++y)

{ ++x; }

printf("%d %d\n",x, y);

}

  a)0 0

b)1 0

c)2 0

d)2 1

* The correct answer is b. x++ translates to 0 (and then x is incremented to 1). Since FALSE && (anything) would become zero, the second part is not evaluated (short-circuited). And overall condition is false and hence ++x is not executed.

19.What is the output of following program?

main()

{

int x = 0;

int y = 0;

if (++x || ++y)

{

x++;

}

printf("%d %d\n",x, y);

}

a)1 1

b)1 0

c)2 1

d)2 0

* The correct answer is d. ++x translates to 1. Since (TRUE || anything) would be TRUE, the second part is not evaluated. Since overall condition is successful, x++ is executed.

20.What is the output of following program?

main()

{

int x = 0;

int y = 0;

if (!(++x || ++y))

{

x++;

}

printf("%d %d\n",x, y);

}

a)1 0

b)1 1

c)2 1

d)2 0

* The correct answer is a. ++x translates to 1. The second part need to be evaluated as (TRUE || anything) would be TRUE. Since there is an overall NOT, condition translates to FALSE, x++ is not executed.

21. What is the output of following program?

main()

{

int x = 10;

int y = 20;

if (x <= y == 1)

{

++x;

}

printf("%d\n", x);

}

a)11

b)10

c)Compiler Dependent

d)Syntax Error

Sorry, the correct answer is a. Precedence: <= followed by ==. The given expression is equivalent to (x <= y) == 1.

22. What is the output of the following program:

#include <stdio.h>

int i=0;

main()

{

int i=5;

printf("%d\n", i++);

}

a)0

b)1

c)5

d)6

* Correct answer is c. Though i is a global variable, the local value of i holds within the block in which it is declared. So printf would print 5, and then a post-increment would take place.

23. Assuming that starting address is 1000, what would be the output?

# include <stdio.h>

main ()

{

int array[] = {2,4,5,6,7};

printf ("%d %d", array,sizeof(array));

}

Answer:

1000, 20

(Note: 4 bytes for int in my compiler)

24. **What is output of following program?**

**int y = 10;**

**main()**

**{**

**int x = 10;**

**int y = 20;**

**x = x + y;**

**if (x >= 30)**

**{**

**int y = 30;**

**x = x + y;**

**}**

**else**

**{**

**int y = 40;**

**x = x + y;**

**}**

**printf("%d\n", x);**

**}**

**a)40**

**b)50**

**c)60**

**d)70**

* **Sorry, the correct answer is c. To begin with, value of x is 10. After x = x + y, it value becomes 30. Control will then get into the if block since x is 30. Within the if block, value of y is 30 and hence x becomes 60.**