#include<stdio.h>

int main()

{

int n;

for(n = 7; n!=0; n--)

printf("n = %d", n--);

getchar();

return 0;

}

Ans:

Output:Above program goes in infinite loop because n is never zero when loop condition (n != 0) is checked.

#include<stdio.h>

int main()

{

   printf("%x", -1<<1);

   getchar();

   return 0;

}

A:

Output is dependent on the compiler. For 32 bit compiler it would be fffffffe and for 16 bit it would be fffe.

# include <stdio.h>

# define scanf  "%s hi hi "

main()

{

   printf(scanf, scanf);

   getchar();

   return 0;

}

Ans:

%s hi hi hi hi

#include<stdio.h>

int main()

{

static int i=5;

if(--i){

main();

printf("%d ",i);

}

}

Ans:

Output: 0 0 0 0  
Explanation: Since i is a static variable and is stored in Data Section, all calls to main share same i.

The first thing you have to remember is that **static variables are initialized only once**.

The second thing is that **static variables have a life time scope and they retain their value between function calls.**

**i**is first initialized to **5**. In theif condition the value of i is changed to**4**. main() is called again and the value of i is changed to **3** in the if condition and main is called again. Now the value of i is changed to **2** and main is called again. Now the value of i is changed to **1**and main is called again. After this the value of i is changed to **"0"** and the block is excited.

As the value of i is now **0**, it is printed 4 times for each of the calls for main().

It compiles because main() is just a function like any other function in the code.

So the ans will be **0 0 0 0 .**

For your clear Explanation -

1. It will expand **as** follows :
2. **void** main()
3. {
4. **int** i = 5
5. **if**(--i) //evaluates to 4
6. { // First recursion
7. **if**(--i) //evaluates to 3
8. { //Second recursion
9. **if**(--i) //evaluates to 2
10. { //Third recursion
11. **if**(--i) //evaluates to 1
12. { //Fourth recursion
13. **if**(--i) //evaluates to 0)
14. { //Fifth recursion
15. //Does not execute.
16. }
17. printf("%d", i); //i=0, so prints 0
18. } //Fourth recursion
19. printf("%d", i); //i=0, so prints 0
20. } //Third Recursion
21. printf("%d", i); //i=0, so prints 0
22. } //Second recursion
23. printf("%d", i); //i=0, so prints 0
24. } //First recursion
25. }

#include<stdio.h>

int main()

{

int i=5;

if(--i){

main();

printf("%d ",i);

}

}

Ans:

Segmentation Fault

#include<stdio.h>

int main()

{

static int var = 5;

printf("%d ",var--);

if(var)

main();

}

Ans:

5 4 3 2 1

# include <stdio.h>

int main()

{

int i=0;

for(i=0; i<20; i++)

{

switch(i)

{

case 0:

i+=5;

case 1:

i+=2;

case 5:

i+=5;

default:

i+=4;

break;

}

printf("%d ", i);

}

getchar();

return 0;

}

Ans:

Explanation:  
Initially i = 0. Since case 0 is true i becomes 5, and since there is no break statement till last statement of switch block, i becomes 16. Before starting the next iteration, i becomes 17 due to i++. Now in next iteration no case is true, so execution goes to default and i becomes 21.

In C, if one case is true switch block is executed until it finds break statement. If no break statement is present all cases are executed after the true case. If you want to know why switch is implemented like this, well this implementation is useful for situations like below.

switch (c)

{

case 'a':

case 'e':

case 'i' :

case 'o':

case 'u':

printf(" Vowel character");

break;

default :

printf("Not a Vowel character");; break;

}

#include <stdio.h>

int main()

{

printf("%p", main);

getchar();

return 0;

}

What would be the output?

Ans:

Address of main fn

#include <stdio.h>

int main()

{

int i;

i = 1, 2, 3;

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

getchar();

return 0;

}

Ans:

Output: 1  
The above program prints 1. Associativity of comma operator is from left to right, but = operator has higher precedence than comma operator.  
Therefore the statement i = 1, 2, 3 is treated as (i = 1), 2, 3 by the compiler.

#include <stdio.h>

int main()

{

int i;

i = (1, 2, 3);

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

getchar();

return 0;

}

Ans:

i = 3

#include <stdio.h>

int main()

{

int first = 50, second = 60, third;

third = first /\* Will this comment work? \*/ + second;

printf("%d /\* And this? \*/ \n", third);

getchar();

return 0;

}

The output is 110 /\* And this? \*/

Is this correct?

Why?

Ans:

Explanation: Compiler removes everything between “/\*” and “\*/” if they are not present inside double quotes (“”).