**Easy Questions**

**1.**

What will be the output of the C program?

#include<stdio.h>

int main()

{

int i;

for(i = 0;i<=3;i++);

printf("%d", (i\*((i\*i)/i)));

return 0;

}

[A. Compilation error](javascript:void(0);)

B. 1 4 9

C. 9

[D. 0 1 2 3](javascript:void(0);)

E 16

Ans 16

**2.**

**24.** What will be the output of the C program?

#include<stdio.h>

int main()

{

int i;

for(i = 0;i<0,5;i++)

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

return 0;

}

[A. 1 3](javascript:void(0);)

[B. Program never ends](javascript:void(0);)

[C. 1 3 5](javascript:void(0);)

[D. None of the above](javascript:void(0);)

Option: B

### Explanation

Here we do not use any brackets in conditional place of for loop, thus compiler will consider that two conditions are given in this place, thus   
1. i< 0 // fails for the first iteration itself   
1. 5 // which is infinite as 5 neven becomes 0.

**3.**

1. What is the output of this C code?

1. #include <stdio.h>
2. printf("%.0f", 2.89);

a) 2.890000  
b) 2.89  
c) 2  
d) 3  
View Answer

Answer: d  
Explanation: None.

**Medium questions**

Given the following input Give the corresponding output

Input 1 = toomuchtohandle

Input 2 = gotcha

#include <stdio.h>

#include <stdlib.h>

int main()

{

int i=0,j=0;

char ch,hc;

while((ch = getchar()) != EOF && i < 14)

{

putchar(ch);

i++;

if(ch == '\n')

putchar('\n');

};

while((hc = getchar()) != EOF && j < 4)

{

putchar(hc);

j++;

if(ch == '\n')

putchar('\n');

};

return 0;

}

Ans

toomuchtohandl

got

What will be the output of the C program?

#include<stdio.h>

int main()

{

int fun = {

printf("toomuchfun ")

};

int x = 5;

for(x=0;x<=fun;x++)

{

printf("%x ", x);

}

return 0;

}

[A.toomuchfun 0 1 2 3 4 5 6 7 8 9](javascript:void(0);)

[B. toomuchfun 0 1 2 3 4 5 6 7 8 9 a](javascript:void(0);) b

[C. 0 1 2 3 4 5 6 7 8 9 a](javascript:void(0);) b

[D. None of the above](javascript:void(0);)

Option: B

### Explanation

for loop executes first.  
for(x=0;x<=fun;x++)  
ie) for(x=0; x=printf("toomuchfun"); x++)  
Now printf in conditional place of for loop prints C for loop and the count in that string is replace in conditional place of for loop   
ie) for(x=0; x<=10;x++)   
thus outputted C for loop 0 1 2 3 4 5 6 7 8 9 a  
here a represent 10 in hexadecimal.

# C program to print message without using any semicolon in program.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | /\*Program to print "Anticoding if fun" (with double quotes) or any message  without using any “;” in program\*/    #include <stdio.h>  int main()  {      if(printf("\”Anticoding is FunC\”\n"))      { }  }  **4.**  .How will you generate infinite loop without     using any loop?  By using **main** we can able to implement infinite loop as i shown below..    #include<stdio.h>  **void main()**  {  printf("Hello");  **main();**  }  Or  main()  {            abc :             if(condition)                 goto bcd:            statements;            goto abc;           bcd:  } |

**Hard questions**

**1.**

# Program to check number is whether EVEN or ODD without using any arithmetic or relational operators.

We can check whether an integer number is **EVEN or ODD without using any Arithmetic or Relational operators**.  
Here is a simple **trick** that we can use to check whether number is **EVEN or ODD**. Using **Logical AND (&) operator** we can check it, each **EVEN number has 0th bit LOW (0) and ODD number has 0th bit HIGH (1)**.  
The statement (number & 0x01) will check that **0th bit is HIGH**, if it is **HIGH (1) number will be an ODD** otherwise **number will be an EVEN**.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | #include <stdio.h>    int main()  {      int number;        //input an integer number      printf("Please input an integer number: ");      scanf("%d",&number);        //check 0th bit of number is 1 or 0      (number & 0x01) ? printf("%d is an EVEN Number.”, number) :  printf("%d is an ODD Number.",number) ;        printf("\n";)      return 0;  } |

**2.**

### Register variables

What is the output of this program?

#include <stdio.h>

int main()

{

register int foo;

printf("%p\n", &foo);

return 0;

}

The register keyword hints to the compiler that the automatic variable should be stored in a register rather than memory, making access to the variable faster. This is only a hint so the compiler is free to ignore it.

In the example we take a pointer to the variable. However, we declared this variable to be stored in a register. Addresses only point to locations in memory so registers can’t be addressed by a pointer. While the compiler can ignore the optimization hint and provide an address, this is ultimately an inconsistent request. The compiler will produce and error and the code will not compile.