18/07/25

50 Logical Reasoning Questions in C programming.

Section 1 : Output Prediction.

1.
```
int a=5;
Printf("%d", a++ +++a);
```
Ans:- 12.

2.
```
int a=10;
if(a=5)
    Printf("True");
else
    Printf("false");
```
Ans : True.

3.
```
int x=10;
Printf("%d", x==10?100:200);
```
Ans:- 100.

4.
```
int a=5, b=5
if(a++ == ++b)
    printf("Equal");
else
    printf("Not equal");
```
Ans:- Equal.

Not equal.

2 | 5 -- 1
2 | 1
0011

out ← value 0000 0011
out from left side. 0000 0110
↳ 6.

5.
```
int x=3;
x = x<<1;
printf("%d", x);
```
Ans:- 6.

6.
```
int a = 10;
int b = a++ + ++a;
Printf("%d", b);
```

7. int arr[] = {1, 2, 3, 4}; to using base pointer Arithmetic

printf("%d", *(arr + 2));

Ans: 3.

8 int x = 0;

while

printf(x < 3)

printf("%d", x++);

Ans:- 0 1 2.

Iteration 1.

0 < 3 ✓

0 1++

1 < 3

1 ✓ 1++

2 < 3 ✓

2 ✓ 2++

3 < 3 ✗

Print 0 1 2

9. int x = 2;

printf("%d", x++ * x++);

4 * 3 = 12

3 * 3 = 9.

10. int a = 5;

int b = 10;

printf("%d", a > b ? a : b);

5 > 10

Ans: 10.

11. char str[] = "Hello";

printf("%c", *str);

Ans:- H

↳ This format specifier only Print the character

12. int a = 1;

int b = 1;

if (a-- && b++)

printf("yes");

else

printf("No");

13. int a = 5;

printf("%d", sizeof(a++));

1 print

15. int a=5;
int *p=&a;
*p = *p + 1;
Printf("%d", a).

Ans: 6.

Section-2. Code Logic Understanding.

1. int x = 10
if (x = 10)
Printf("Yes").

Ans:- Yes.

Here the condition is assing x = 10 only not to compare that's why. yes.

2. const int x = 10;
x = 20.

x value not able to change. it declare in the const declaration
Output :- compile time error.

Ans: 10.

23.01.25

3. int *ptr;
*ptr = 5;
in right way.

->*ptr = 5 try to write a random memory to happen
-> segmentation fault
-> overwriting critical memory
-> unpredictable behaviour

int a;
int *ptr = &a;
*ptr = 5
Printf("%d", a); O/P = 5.

4. can a function return an array in c?

In function not able to return array.

because. cannot return arrays directly like Primitive type. (when declared inside a function, they exist in stack memory).

If function return array it is possible in few ways.

↳ use static array. ↳ use dynamic array
↳ pass an array to the function and fill it inside

Ans:-

function cannot return an array directly because arrays are not returnable types and local arrays go out of scope after the function ends. to achive pointer to static array, dynamically allocated array.

5. int a = 5;

{

int a = 10;

}

Ans:- int a = 5;

{ int a = 10;

printf("%d", a);

}

printf("%d", a);

Ans 5 10.

6. what happens if you access an array out of bound?

Accessing an array out of bounds is a undefined behaviour. It may crash, corrupt memory silently fail. C does not check the array bounds.

Ans:-

10

7. int i;

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

printf("%d", i);

8. int a = 5;

printf("%d", ++a++);

Invalid error or

9. int a=1, b=2;

int c = a++ + b;

1 0

a++ + b

1 + 2 = 3

10. static keyword?

Inside a function, a static variable retains its value between function calls instead of being reinitialized every time.

Ans:- b.

11. int i=0;
while (i++ < 5)
    continue;
printf("%d", i);

i	i++ < 5	Print i
0	0 < 5 then i=1	1
1	1 < 5 i=2	2
2	2 < 5 i=3	3
3	3 < 5 i=4	4
4	4 < 5 i=5	5
5	5 < 5 false	i=6

12). What is NULL?

NULL is macro that represents a null pointer. A null pointer is a special pointer that does not point to any valid memory location. It helps to avoid undefined behaviour from using uninitialized or dangling pointer.

Ex:- After memory allocation with malloc, if allocation fails, the returned pointer is NULL,
the pointer doesn't point anywhere yet.

13). To write a correct to declare a function returning pointer to int?

#include <stdio.h>
int *get_num() {
    static int num = 10;
    return #
}
int main()
{
    int *ptr = get_num();
    printf("%d", *ptr);
    return 0;
}

&num - the address of num that stored in the *ptr.

O/P:- 10.

14) When does the segmentation fault occur in C?

A segmentation fault occurs when a program tries to access restricted memory. such as dereferencing a NULL or invalid pointer, writing a outside of array bounds, modifing read only memory. It is a runtime error that usually crashes the program.

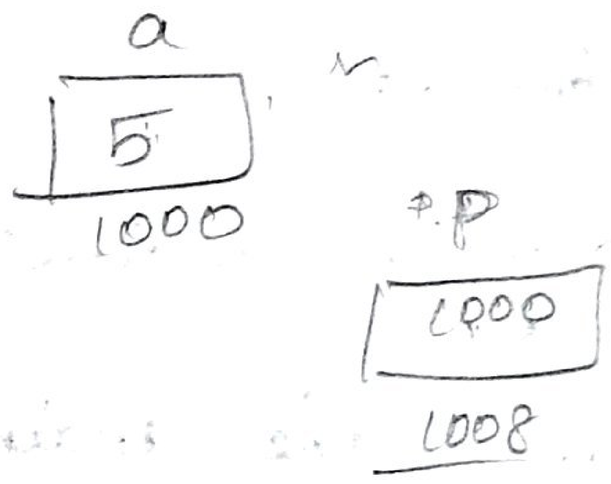
15). can a void function return a value?

NO, a void function cannot return a value.

Section -3

1. int a = 5;
   int *p = &a;
   printf("%d", *p);

Ans: 5

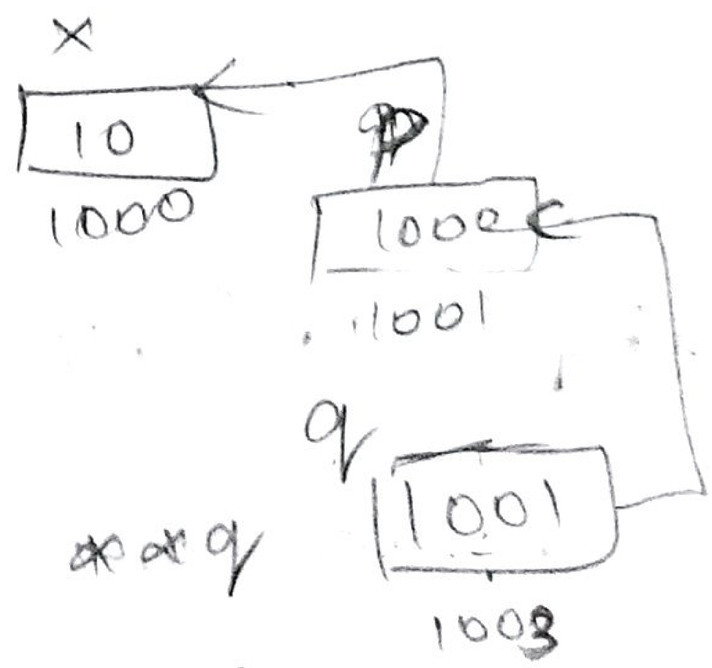


2. int *ptr = NULL; → It returns the segmentation fault
   printf("%d", *ptr);

   NULL → means does not point any valid memory address.

   *ptr → it is trying to dereference the a null pointer.

10. int x = 10
    int *p = &x;
    int **q = &p;
    printf("%d", **q); // 10.



Dynamic Memory Allocation :-

SMA → at compile time

DMA → at run time.

↓

→ the memory allocated & deallocated at run time.

↳ malloc()

↳ calloc()

↳ realloc()

↳ free()

Section – 4 :-

1. Why is main() special in C ?

The main() function in c special because it serves as the entry point for program execution. This means that when a c program is compiled and executed, the operating

2. What is the output type of 'sizeof()' ?

the exact number it returns depends on the data type and the platform / compiler.

3. Printf("%d", sizeof('A'));

In this case character constant like 'A' are of type int, not char. So sizeof('A') returns the sizeof an int, typically 4 bytes. & It gives the format specifier warning :-

Printf("%zu", sizeof('A'));

4. What are the header files used for?

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <stdlib.h>
```

5. Difference between '==' and '=' in c?

'=' is the assignment operator. It stores a value into a variable.

'==' is the comparison operator, it checks if two values are equal.

6. What does 'volatile mean?

the volatile keyword is used to tell the compiler.

"Do not optimize this variable - its value may change unexpectedly at any time".

8. what happens when 'break' is used in a loop?

The break statement is used to immediately exit a loop, even if the loop condition hasn't become false. When the break is encountered the inside the loop like for, while or do-while, the control jumps out of the body and continues with the next statement after the loop.

7. Function pointer?

A function pointer is a variable that stores the memory address of a function, allowing you to call the function indirectly through the pointer.

A function pointer is a pointer that points to a function instead of pointing to a variable. It stores the address of a function, we can use it to call the function indirectly.

function pointer are useful when we want to pass a function as an argument, implement callbacks, or write plugh-like code.

8. Difference between 'int func()' and 'int func(void)'

int func() → It returns the integer value

int func(void) → It returns nothing, because the passing void data type.

Precedence high to low

10. int a=1;
    int b=0;
    Printf("%d", a&& b || !a);

!
&&
||

=> 0001 && 0000 || !(0001)

=> 0001 && 0000 || 1110

=> 0000 || 1110 => ||

0000
1110
1110

AND (!)

= (00001 && 0000) || !(1);

= 0000 || 0

24.07.25

Dynamic Memory Allocation :-

Dynamic allocated memory using pointer.

malloc (sizeof (int));

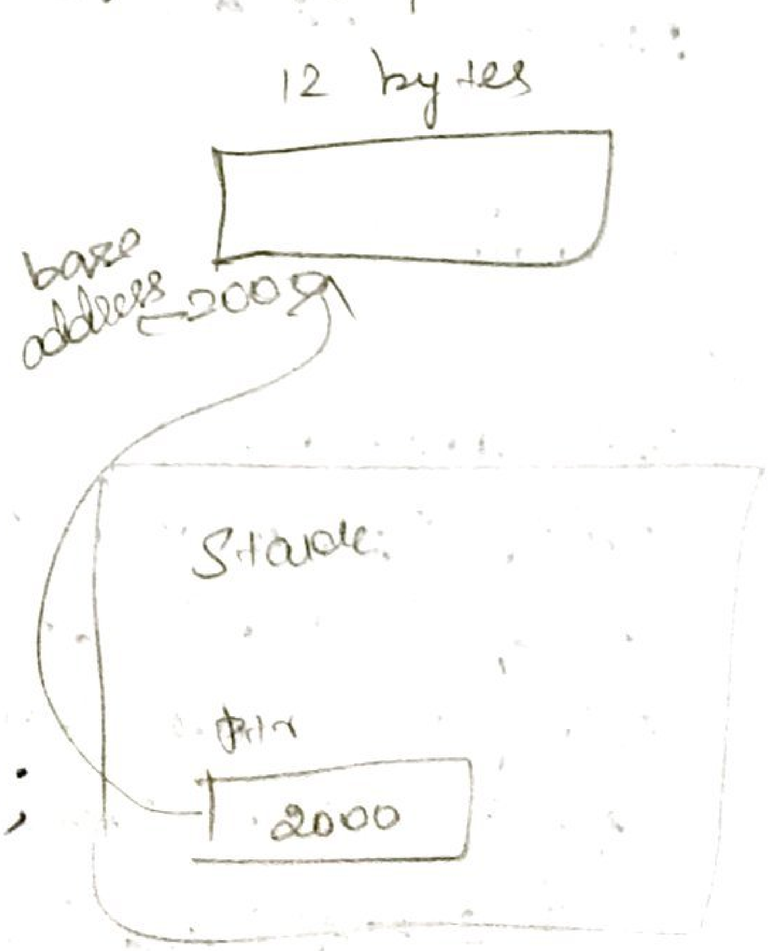
malloc return type is void pointer.

because here to store any type of value through the typecast.

ptr = (int *) malloc (3 * sizeof (int));

ptr = (int *) malloc (n * sizeof (int));

```
int main()
{
  int n, i, *ptr;
  printf ("Enter the total number values");
  scanf ("%d", n);
  ptr = (int*) malloc (n * sizeof (int));
  printf ("Enter the values");
  for ( i=0; i<n; i++)
    { scanf ("%d", (ptr+i));
    }
  printf ("the entered values");
  for (i=0; i<n; i++)
    { printf ("%d", *(ptr+i));
    }
  free (ptr);
}
```



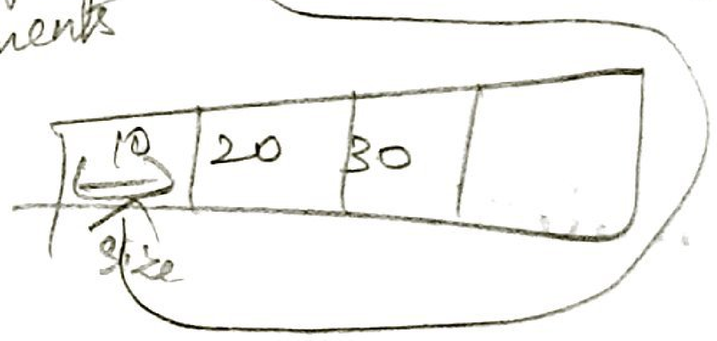
calloc :-

↳ continuous allocation.
↳ built in function stdlib.h
↳ used to dynamically allocate multiple block of memory & each block is of same size.

Once the memory would be allocated to initialize zero.

```
ptr = (int *) calloc (n, sizeof (int));
```

↳ count of size of elements
↳ size of each block



Realloc :-

realloc → resize increase / decrease
→ reallocation.

```
ptr1 = (int *) realloc (ptr, 7 * sizeof (int));
```

↳ old pointer
↳ to decide incr./decr the memory size.

```
int main()
{
  int *ptr, n, i;
  printf ("Enter value of n");
  scanf ("%d", &n);
  printf ("Enter the values");
  for(i=0; i<n; i++)
    { scanf("%d", (ptr+i)); }
    }
```

```
ptr = (int *) malloc (n * sizeof (int));
printf ("Enter the values");
for (i=0; i<n; i++)
{
    scanf ("%d", (ptr+i));
}
printf ("Enter updated size of n:");
scanf ("%d", &n);
int *ptr1 = (int *) realloc (ptr, n * sizeof (int));
printf ("previous address: %d, new address: %d",
                                    ptr, ptr1);
printf ("the entered values");
for (i=0; i<n; i++)
  {  printf ("%d \t", *(ptr1+i));
  }
free (ptr1);
}
```

3). what is the difference between malloc() & calloc()

malloc() and calloc() are both used for dynamic memory allocation in c. the key different is that malloc() only allocates memory and leaves it uninitialized, which means it contains garbage values. In calloc() not only allocates memory but also initializes every byte to zero. malloc() - is single size argument and the calloc() is two argument — the number elements and size of each.

4). what does free() do?

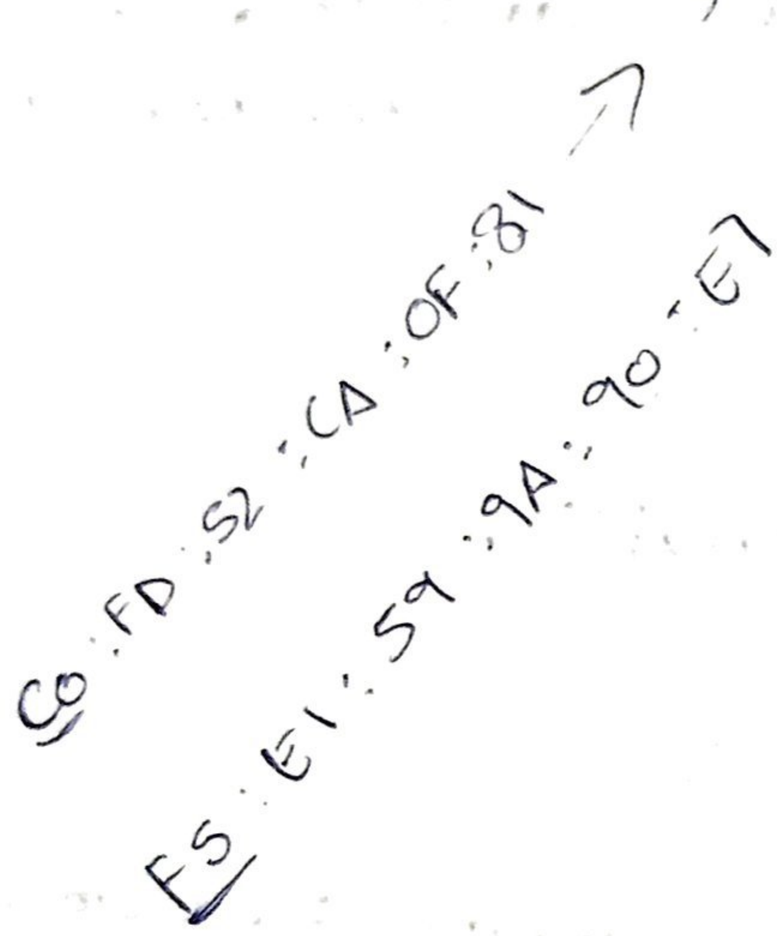
free() deallocates dynamically allocated memory leaks. After calling it, you should not use the pointer again unless it is reassigned.

5. int *P;
   *P = 10;

⇒ undefined behaviour.

In the code the pointer P is uninitialized and points to an unknown location. Dereferencing it and assigning 10 the result is undefined behaviour. usually crash.

Sanjeet



6. Can a pointer point to a function?

Yes, the pointer can point to a function. This is called a function pointer. It can be used to call functions directly, pass them as argument or implement callbacks. It's especially

7. Dag. Dangling pointer :-

A dangling pointer is a pointer that points to memory that has been freed, deleted, out of scope meaning the memory it refers to is no longer valid.