Q1) What is a Pointer? Explain with example.

Answer:

A pointer is a special variable that stores the **memory address** of another variable instead of storing a value directly.

Example:

```
int x = 10;

int *p;

p = &x;
```

Here:

- p is pointer variable
- p stores address of x
- *p represents value stored at that address (10)

Pointers improve efficiency, allow dynamic memory handling and support structures, arrays and functions.

Q2) What is the use of address operator & and dereference operator *?

Answer:

Address operator (&) gives memory address of a variable.
Example: &x gives address of x.

Dereference operator (*) accesses the value stored at address held by pointer.

Example:

```
int x=10;

int *p=&x;

printf("%d", *p);  // prints 10
```

Q3) What is NULL pointer and void pointer?

NULL Pointer:

Pointer which does not point anywhere.

Example:

```
int *p = NULL;
```

Used to safely indicate that pointer is empty.

Void Pointer:

Pointer that can store address of **any data type**.

Example:

```
void *p;

int x=10;

p=&x;

printf("%d", *(int*)p);
```

Typecasting required before dereferencing.

Q4) Explain pointer arithmetic with examples.

Answer:

Pointer arithmetic means performing operations like:

pointer + integer
pointer – integer
pointer++ / pointer--

Example:

```
int a[3]={10,20,30};

int *p=a;
```

```
p++;      // now points to 20
printf("%d", *p); // prints 20
```

Pointer movement depends on **data type size**, not 1 byte.

Q5) What is pointer to pointer? Explain with an example.

Answer:

Pointer to pointer stores the **address of another pointer**.

Example:

```
int x=10;
int *p=&x;
int **pp=&p;
printf("%d", **pp); // output: 10
```

Useful in dynamic structures and multi-level memory referencing.

Q6) How are arrays and pointers related?

Answer:

Array name works like pointer to its first element.

Example:

```
int a[5]={10,20,30,40,50};
int *p=a;
printf("%d", *p);     // 10
printf("%d", *(p+1)); // 20
```

Equivalent forms:

```
a[i] == *(a+i)
p[i] == *(p+i)
```

Q7) What is an array of pointers? Provide example.

Answer:

An array storing **addresses** instead of values.

Example (string list):

```
char *names[3] = {"Ram","Amit","Atul"};
```

Each element holds address of a string.

Q8) What is pointer to a function? Give an example.

Answer:

Pointer that stores address of function.

```
int add(int a,int b)

{

   return a+b;

}

int (*fp)(int,int)=add;

printf("%d", fp(3,4));   // calls add(3,4)
```

Useful in callback and runtime decision logic.

STRUCTURE BASED QUESTIONS

Q9) What is a structure? Why is it needed?

Answer:

A structure is a **user-defined datatype** that groups variables of **different data types** under one name.

Example: student record can hold name (string), roll (int), marks (float).

Structures allow:

grouping related values
better data modeling
storage of complex records

Q10) How do you declare and access structure members?

Answer:

Declaration:

```
struct student {

    int roll;

    char name[20];

    float marks;

};
```

Structure variable:

```
struct student s1;
```

Accessing members (dot operator):

```
s1.roll = 1;

strcpy(s1.name, "Atul");

s1.marks = 85.5;
```

Q11) Write a program to read and display information of 3 students using structure.

```
#include<stdio.h>

struct student {

    int roll;

    char name[20];
    float marks;
};
void main()
{
    struct student s[3];
    int i;
    for(i=0;i<3;i++){
        printf("Enter roll, name, marks: ");
        scanf("%d %s %f",&s[i].roll, s[i].name, &s[i].marks);
    }
    printf("\nStudent Details:\n");
    for(i=0;i<3;i++){
        printf("%d %s %.2f\n",s[i].roll, s[i].name, s[i].marks);
    }
}
```

Q12) What is nested structure? Provide example.

Answer:

Structure inside another structure.

Example:

```
struct date {
    int day, month, year;
};
```

```
struct employee {
    char name[20];
    struct date joining;
    float salary;
};
```

Access:

```
emp.joining.day = 10;
```

Q13) What is array of structures? Give example.

Answer:

Structure variables stored in array form.

```
struct student {
    int roll;
    char name[20];
};
struct student s[20]; // 20 student records
```

Used for maintaining bulk records.

Q14) Explain structure with function.

Passing structure by value:

```
void display(struct student s)
{
    printf("%s %d %.2f", s.name, s.roll, s.marks);
}
```

Passing structure using pointer:

```
void display(struct student *p)

{

   printf("%s %d %.2f", p->name, p->roll, p->marks);

}
```

Pointer method is efficient for large structures.

Q15) What is structure pointer? How do you access members?

Answer:

Pointer storing address of structure variable.

Example:

```
struct student s1;

struct student *p;

p=&s1;

p->roll = 10;          // same as (*p).roll
```

-> operator is used to access members via pointer.

UNION BASED QUESTIONS

Q16) What is a Union? How is it different from a structure?

Answer:

Union is a user-defined datatype where **all members share the same memory location**, so **only one member is valid at a time**.

Differences:

Structure	Union
Each member has separate memory	All members share same memory
Size = sum of all members	Size = largest member
All members valid	Only one member valid at a time

Q17) Declare a union and explain its memory behavior.

Answer:

```
union data {

   int i;

   float f;

   char ch;

};
```

If largest member is float (say 4 bytes),
size of union = 4 bytes only.

Writing into one member overwrites the others.

COMBINED EXAM STYLE BIG QUESTION

Q18) Explain different types of pointers used in C.

✔ **Answer:**

1. **Normal Pointer**
   Stores address of variable
   int *p = &x;
2. **NULL Pointer**
   Does not point anywhere
   int *p = NULL;
3. **Void Pointer**
   Generic pointer, can store address of any type
   void *p = &x;
4. **Pointer to Pointer**
   Stores address of another pointer
   int **pp = &p;
5. **Array Pointer**
   Array name stored as pointer
   int *p = a;
6. **Function Pointer**
   Stores function address
   int (*fp)(int,int)=add;
7. **Structure Pointer**
   Pointer storing address of structure variable
   struct student *p = &s;

Q19) Write a C program to swap two values using pointers.

```
#include<stdio.h>

void swap(int *x, int *y)

{

   int temp;

   temp = *x;
```

```
    *x = *y;
    *y = temp;
}
void main()
{
    int a=5, b=10;

    swap(&a, &b);
    printf("After swap: a=%d, b=%d", a, b);
}
```