CHAPTER NO: 5	POINTER
1	What is pointer? What is the use of it?
2	How can we define pointer? Explain with simple example.
3	Explain array of pointer with example.
4	What are advantages and disadvantages of pointer?
5	In pointer arithmetic which operations are invalid and which are valid?
6	Explain pointer to pointer with example.
7	Demonstrate use of pointer with string.

Q-1 WHAT IS POINTER? WHAT IS THE USE OF IT?

ANS:

DEFINITION OF POINTER:

- Pointer is a derived data type in C that contains memory addresses of the variables.
- Memory addresses are the locations in the computer memory where program instructions and data are stored.

USE OF POITNER:

- Pointers are just like the simple variables but pointer contains the address of other variables and so pointers can be used to access and manipulate the data stored in memory.

Q-2 HOW CAN WE DEFINE POINTER? EXPLAIN WITH SIMPLE EXAMPLE.

ANS:

- The general form of pointer declaration :

  datatype *pointer_variable;

- The asterisk (*) tells that the pointer_variable is pointer variable.

Example:

```
int *p;
```

- This statement declares the variable p as a pointer variable that points to an integer data type.

```
#include<stdio.h>
#include<conio.h>

void main()
{
        int a;
        int *p;
        clrscr();
        p=&a;
        printf("\n %u",p);
        printf("\n %d",*p);
        getch();
}
```

Q-3 EXPLAIN ARRAY OF POINTER WITH EXAMPLE.

ANS:

- **Considering 1-d array:**

  Syntax: **datatype arrayname[size];**

  Example: int a[5];

- When an array is declared, the compiler allocates a base address and sorage to contain all the elements of the array in memory locations.
- The base address is the location of the first element (index 0) of the array.
- The compiler defines the array name as a constant pointer which contains the address of the first element of the array.
- If p is a pointer to an integer, then declared as
  int *p;
- p=&a[0] point to the element zero of a; that is p contains the address of a[0]. It means that the address of the first array element can be expressed as either &a[0] or simply a.
- Now the assignment x=*p will copy the contents of a[0] into x.
- If p points to particular element of array, then by definition p+1 points to the next element.

p	p+1	p+2	p+3	p+4
a[0]	a[1]	a[2]	a[3]	a[4]
1000	1002	1004	1006	1008

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[5]={10,11,12,13,14};
        int *p;
        clrscr();
        p=&a[0];
        printf("\n %u",p);
        printf("\n %d",(p+1));
        printf("\n %u",(p+2));
        printf("\n %d",(p+3));
        printf("\n %d",(p+4));
        getch();
}
```

Q-4 WHAT ARE ADVANTAGES AND DISADVANTAGES OF POINTERS?

ANS:

ADVANTAGES:

1) Pointers allow you to implement pass by reference (pass by address).
2) Pointer allows the modifications by a function that is not the creator of memory.
   that is function A() can allocate memory and function c() can modify it.

```
e.g.  void A(int x)          void B(int *p)
      {                      {
            B(&x);                 *p=*p+1;
      }                      }
```

3) Pointer allows you to use dynamic memory allocation.
4) Pointer gives the ability to implement complex data structures like linked list, trees, etc.
5) Pointer allows ease of programming especially when you are dealing with strings. pointer is incremented as per the size of the variable, that is if the variable is integer then pointer is incremented or decremented by 2 bytes.

Example:

```
#include<stdio.
#include<conio.h>

void main()
{
    int *p;
    int a;
    p=&a;
    printf("%u",p);
    p++;
    printf("\n %u",p);
    getch();
}
```

6) Pointer allows you to resize the data structure whenever needed. For e.g. if you have an array of size 10 it can not be resized but array created using pointer can be resized.

DISADVANTAGES:

1) If sufficient memory is not available during runtime for the storage of pointers, the program may crash (leas possible).
2) If the programmer is not careful and consistent with the use of pointers, the program may crash (very possible).

Q-5 IN POINTER ARITHMETIC WHICH OPERATIONS ARE VALID AND WHICH ARE INVALID?

ANS:

VALID POINTER ARITHMETIC OPERATIONS:

1) Pointers can be incremented or decremented according to the size of the variable to which it points.

Example:

```
int *p;
int a;
p=&a;
printf("%u",p);
p++;
printf("\n %u",p);
```

2) If p1 and p2 are properly declared and initialized, the following operations are valid:

```
a=a+*p1;
*p1=*p1+1;
mul=*p1 *  *p2;
div=*p1 /  *p2;
```

- In the last statement, blank space must be kept between / and * because if we write /* then it will be considered as beginning of the comment.

3) We can also use expressions like p1==p2, p1<p2, p1!=p2.

4) Pointer variable can be assigned null value.

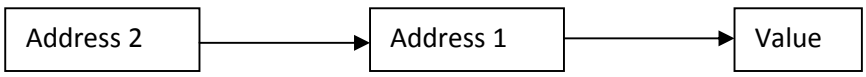
INVALID POINTER ARITHMETIC OPERATIONS:

1) Two pointers are never added.
2) We can not use multiplication and division of a pointer variable with constant.
3) An invalid pointer reference occurs when a pointer's value is referenced even though the pointer doesn't point to a valid block. Suppose p and q are two pointers. If we say p=1 and q is uninitialized. The pointer p will then become uninitialized as well, and any reference to *p is an invalid pointer reference.

Q-6 EXPLAIN POINTER TO POINTER WITH EXAMPLE.

ANS:

- In C programming, it is also possible that pointer variable is also pointed by another pointer variable. That process is known as pointer to pointer.



- Here, the pointer variable p2 contains the address of the pointer variable p1, which points to the location that contains the desired value. This is known as multiple indirection.

- A variable that is pointer to pointer must be declared using additional indirection operator (*) symbols in front of the name.

Example:

```
int **p2;
```

- This declaration tells the compiler that p2 is a pointer to a pointer of int type.

Example:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int x,*p1,**p2;
        x=100;
        p1=&x;
        p2=&p1;
        printf("\n %d",**p2);
        getch();
}
```

Q-7 DEMONSTRATE USE OF POINTER WITH STRING.

ANS:

- String is an array of characters ending with additional null character ('\0') which specifies the termination of string in memory.
- Pointer array offer a particularly convenient method for storing strings. In this situation, each array element is a character type pointer that indicated the beginning of separate string.
- Each individual string can be accessed by referring to its corresponding pointer.
- For example, char name[10][12]; here name contains 10 rows to accommodate 10 strings. Each row must be large enough to store at least 12 characters. A better way to do this is to define a 10 element of pointer, that is

  char *name[10];
- Thus, the name[0] will indicate the first name, name[1] will indicate the second name and so on.

Example:

```
#include<stdio.h>
void main()
{
        char *message[8]={“Four”,”score”,”and”,”seven”,”years”,”ago”,”our”,”forefathers”};
        int count;
        clrscr();
        for(count=0;count<8;count++)
        {
                printf(“%s”,message[count]);
        }
        getch();
}
```