

# Subject: Programming in C

## Chapter : Arrays and Pointers

### Topic: Arrays and Pointers – 6

DPP-03

**[NAT]****1.** Consider the following program:

```
#include<stdio.h>
int main()
{
    int p=10, *q;
    q=&p;
    *q=p+++*q;
    printf("%d", *q);
    return 0;
}
```

The output is \_\_\_\_\_.

**[MCQ]****2.** Consider the following program:

```
#include<stdio.h>
int * f(){
    static int a[4]={ 1, 2, 3, 4};
    return a;
}

int main()
{
    int *p, i;
    p=f();
    for(i=0;i<3;i++){
        printf("%d\t", p[i]+p[i+1]);
    }
    return 0;
}
```

The output is-

- (a) Compilation Error
- (b) Runtime Error
- (c) 3 5 7
- (d) None

**[NAT]****3.** Consider the following program:

```
#include<stdio.h>
int main()
{
    int p=10, s=20, *q, **r;
    q=&p;
    *q=p+++*q;
    q=&s;
    r=&q;
    **r=--*q***r;
    printf("%d", p+s);
    return 0;
}
```

The output is \_\_\_\_\_.

**[MCQ]****4.** Consider the following program:

```
#include<stdio.h>
int * f()
{
    int a[4]={ 1, 2, 3, 4};
    return a;
}

int main()
{
    int *p, i;
    p=f();
    for(i=0;i<3;i++){
        printf("%d\t", p[i]+p[i+1]);
    }
    return 0;
}
```

The output is-

- (a) compilation Error
- (b) Runtime Error
- (c) 3 5 7
- (d) None

**[MCQ]**

5. Consider the following statements:

P: `int * p(int *)` - `p` is a function that takes an integer pointer as argument and returns an integer pointer.

Q: `int (*p(int *))[ ]` - `p` is a function that takes an integer pointer as argument and returns a pointer to an array of integers.

Which of the following is INCORRECT?

- |                  |                      |
|------------------|----------------------|
| (a) P only       | (b) Q only           |
| (c) Both P and Q | (d) Neither P nor Q. |

**[MCQ]**

6. Consider the following program:

```
#include<stdio.h>
void f(int (*q)[2]){
    printf("%d\t",(*q)[1]);
    q+=2;
    printf("%d",(*q)[1]);
}
int main()
{
    int a[][2]={2,4,6,8,10,12};
    int (*ptr)[2]=a;
    f(ptr);
    return 0;
}
```

The output is:

- |          |         |
|----------|---------|
| (a) 4 12 | (b) 4 8 |
| (c) 2 10 | (d) 2 6 |

**[MCQ]**

7. Consider the following program:

```
#include<stdio.h>
int main()
{
    int a[3]={0,1,2};
    int *p=(int *)&a+1;
    printf("%d\t%d", *(a+1),*(p-1));
    return 0;
}
```

The output is-

- |                        |
|------------------------|
| (a) Garbage value      |
| (b) Segmentation fault |
| (c) 1 2                |
| (d) Compilation Error  |

**[MCQ]**

8. Consider the following program:

```
#include<stdio.h>
void fun(int n){
    for(n--;;n--n)
        printf("GATE WALLAH");
}
int main()
{
    void (*p)(int)=fun;
    (*p)(6);
    return 0;
}
```

The output is-

- |  |
|--|
| (a) Compilation Error                              |
| (b) Runtime Error                                  |
| (c) printf() is executed infinite number of times. |
| (d) print() is executed two times.                 |

## Answer Key

- |          |        |
|----------|--------|
| 1. (21)  | 6. (a) |
| 2. (c)   | 7. (c) |
| 3. (382) | 8. (d) |
| 4. (b)   |        |
| 5. (d)   |        |



## Hints and solutions

1. (21)

	1000
p	<del>10 20 21</del>
	2000
q	1000

\*q=p+++\*q;=10+\*1000=10+10=20;  
p is then incremented by 1.

Final value at p=21

2. (c)

No error, since it returns the address of static array.

for(i=0;i<3;i++) printf("%d\t", p[i]+p[i+1]);

The loop prints p[0]+p[0+1], p[1]+p[1+1],  
p[2]+p[2+1].

Output: 3 5 7

3. (382)

	1000		3000
p	<del>10 20 21</del>	s	<del>20 19 361</del>
	2000		4000
q	<del>1000</del> 3000	r	2000

p+s = 21 + 361 = 382

4. (b)

Runtime error exists since it returns the address of local array variable.

5. (d)

Both the statements P and Q are CORRECT.

P: CORRECT.

int \*p(int \*) - p is a function that takes an integer pointer as argument and returns an integer pointer.

Q: CORRECT

int (\*p(int \*))[] - p is a function that takes an integer pointer as argument and returns a pointer to an array of integers.

6. (a)

```
void f(int (*q)[2]){
    printf("%d\t",(*q)[1]);//It prints the first element of
    the 0th row of a. So, 4 is printed.
    q+=2;//q now points to the 2nd row of a.
    printf("%d",(*q)[1]);// It prints the first element of the
    2nd row of a. So, 12 is printed.
}
```

Output: 4 12

7. (c)

Suppose the elements 0, 1, 2 are stored at locations  
100, 102, 104 (assuming integer size of 2 bytes).

```
int a[3]={0,1,2};
int *p=(int *)(&a+1);//p contains 106
printf("%d\t%d", *(a+1),*(p-1));/*(100+1)=*102=1
and *(106-1)=*104=2. So, 1 2 are printed.
return 0;
}
```

Output: 1 2

8. (d)

p is a pointer to the function fun.

```
void fun(int n){ //n=6
    for(n--; --n ;--n)
        6 4 -> printf() 3
        2 -> printf() 1
        0->Loop stops
}
```

So, printf() is executed 2 times.



Any issue with DPP, please report by clicking here:- <https://forms.gle/t2SzQVvQcs638c4r5>

For more questions, kindly visit the library section: Link for web: <https://smart.link/sdfez8ejd80if>