CSE/IT   Batch: Hinglish

Subject: Programming in C
Topic: Arrays and Pointers – 2 | DPP-02 |

[MCQ]
1. Consider the following program:

```
#include<stdio.h>
int main()
{
    int a[5]={5, 3, 1, 2, 4};
    int *p[5]={a, a+1, a+3, a+2, a+4};
    printf("%u\t%u", p[3][1], *(*(p+4)-2));
    return 0;
}
```
The output is:

(a) 5 3   (b) 4 1

(c) 2 1   (d) Garbage value.

[MCQ]
2. Consider the following program:

```
#include <stdio.h>
int main()
{
    int a[]={2, 4, 6};
    int b[]={1, 3, 5};
    int *arr[]={a, b};
    printf("%u\t", *(*(arr+1)+2));/*line 1*/
    printf("%u\t",**arr+3);/*line 2*/
    printf("%u",***arr);/*line 3*/
    return 0;
}
```

Which of the following lines does not give ERROR?

(a) Line 1 only

(b) Line 1 and Line 2 only

(c) Line 3 only

(d) Line 2 and Line 3

[MCQ]
3. Consider the following function:

```
void f(int *p, int n)
{
    static int i;
    i=n-1;
    if(i<0) return;
    printf("%d\t", p[i]+p[n-i-1]);
    f(p,n-1);
}
```
If the array arr with elements [1, 2, 3, 4, 5] is passed as f(arr, 5), the output is-

(a) 5 4 3 2 1   (b) 7 6 5 4 3

(c) 6 5 4 3 2   (d) 2 3 4 5 6

[NAT]
4. Consider the following program:

```
#include<stdio.h>
int main()
{
    int a[5]={5, 3, 1, 2, 4};
    int *p[5]={a+3, a+1, a, a+2, a+4};
    int **ptr=p+3;
    printf("%u\t%u\t%u", ptr-p, *ptr-a, **ptr);
    return 0;
}
```
The sum of the output is _____.

5. Consider the following function:

```
void func(int (*ptr) [2])
{
    **ptr+=1;
    ptr++;
    **ptr*=3;
}
```
The array arr[2][2] with elements {0, 1, 2, 3} is passed to func(). What are the contents of the array after calling func()?

(a) 1 1 6 3   (b) 0 1 2 3

(c) 1 1 2 3   (d) Compilation Error.

[MCQ]

6. Consider the following program:

```
#include<stdio.h>
int main()
{
 int a[][3][2]={1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23};
 printf("%u\t", a+1);
 printf("%u\t", *a+1);
 printf("%u\t",**a+1);
 printf("%u\t", ***a+1);
 printf("%u\t", &a+1);
 return 0;
}
```
Assume the base address of a is 100 and integer size is 2 bytes, the output is-

(a) 124 112 102 2 106

(b) 124 102 112 5 106

(c) 112 104 102 2 124

(d) 112 104 102 5 124

[MCQ]

7. Consider the following program:

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

```
int main()
{
   int a[3][2]={1, 3, 5, 7, 9, 11};
   int *ptr=a;
   ptr+=sizeof(int);
   printf("%d", *ptr);
   return 0;
}
```
(Assume size of int to be 2 bytes.)

The output is _____.

8. Consider the following program:

```
#include<stdio.h>

int main()

{
   int a[][2]={1, 3, 5, 7, 9, 11};
   int *ptr=a[1];
   ++*ptr++;
   printf("%d", *ptr);
   return 0;

    }
```
The output is-

(a) 5   (b) 9

(c) 7   (d) Compilation error

Answer Key

1.	**(c)**		**6.**	**(c)**
2.	**(b)**		**7.**	**(5)**
3.	**(c)**		**8.**	**(c)**
4.	**(6)**			
5.	**(a)**			

Hints and solutions

1. (c)

	1000	1002	1004	1006	1008
a	5	3	1	2	4

	2000	2004	2008	2012	2016
p	1000	1002	1006	1004	1008

p[3][1]=*(*(p+3)+1)=*(*2012+1)=*(1004+1)=*1006 =2

((p+4)-2)= *(*2016-2)=*(1008-2)=*1004=1

Output: 2 1

2. (b)

	1000	1002	1004
a	2	4	6

	2000	2002	2004
b	1	3	5

	3000	3004
arr	1000	2000

```
printf("%u\t", *(*(arr+1)+2));//It prints arr[1][2]
i.e 5
```

```
printf("%u\t",**arr+3);//It prints arr[0][0]+3 i.e 5
printf("%u",***arr);// It gives ERROR as here
dereferencing can happen only at two levels
```

3. (c)

	1000	1002	1004	1006	1008
a	1	2	3	4	5

```
f(arr, 5):
p=1000, n=5;
static int i; //i is initialized to 0
i=n-1; //i=4
4<0: FALSE
printf("%d\t", p[i]+p[n-i-1]);//p[4]+p[0]=5+1=6 is
printed.
f(1000, 4) is called.
f(1000, 4):
```

```
i=n-1; //i=3
3<0: FALSE
printf("%d\t", p[i]+p[n-i-1]);//p[3]+p[0]=4+1=5 is
printed.
f(1000, 3) is called.
f(1000, 3):
i=n-1; //i=2
2<0: FALSE
printf("%d\t", p[i]+p[n-i-1]);//p[2]+p[0]=3+1=4 is
printed.
f(1000, 2) is called.
f(1000, 2):
i=n-1; //i=1
1<0: FALSE
printf("%d\t", p[i]+p[n-i-1]);//p[1]+p[0]=2+1=3 is
printed.
f(1000, 1) is called.
f(1000, 1):
i=1-1; //i=0
0<0: FALSE
printf("%d\t", p[i]+p[n-i-1]);//p[0]+p[0]=1+1=2 is
printed.
f(1000, 0) is called. It simply returns.
Output: 6 5 4 3 2
```

4. (6)

	1000	1002	1004	1006	1008
a	5	3	1	2	4

	2000	2004	2008	2012	2016
p	1006	1002	1000	1004	1008

```
ptr=p+3
ptr-p=p+3-p=3
*ptr-a=*(p+3)-1000=*2012-1000=(1004-1000)/2=2
**ptr=**(p+3)=**2012=*1004=1
Output: 3 2 1
Sum: 6
```

5. **(a)**

The function increments the 0th element of 0th row by 1 and 0th element 1st row by 3.

Output: 1 1 6 3

6. **(c)**

a+1 points to the 1st 2D array.
*a+1 points to the 1st 1D array of the 0th 2D array.
**a+1 points to the 1st element of the 0th 1D array of the 0th 2D array.
***a is the the 0th element of the 0th 1D array of the 0th 2D array.
&a+1 is the address of the next 3D array.
Output is:

112 104 102 2 124

7. **(5)**

ptr stores the address of the 0th 1D array.
ptr+=2 means ptr now points to the 2nd element.

printf("%d", *ptr);//5 is printed.

8. **(c)**

	1000	1002	1004	1006	1008	1010
a	1	3	5	7	9	11

ptr is a single pointer. So, it increments linearly.

ptr stores the address of (a+1). ptr=1004;
++*ptr++ increments ptr by 1.
ptr=1006

Output: 7