C Arrays Questions

1. What will be the output of the C program?

```
#include<stdio.h>

int main(void)

{

        int arr[5] = { 1, 2, 3, 5, 7 };

        int *ptr = (&arr + 1);

        printf("%d %d\n", *(arr + 1), *(ptr - 1));

        return 0;

}
```

A. 2 5

B. 3 5

C. 2 7

D. 3 7

x

Option: C

Explanation

let's go from line 5...

*ptr = (address of first value in arr[] array + 1)

let us consider 2293416 is a address of first value in arr[] array i.e) *ptr = (2293416 + 1)

i.e) *ptr = (2293436) and not 2293420 because 1 points to the next location after all the addressess of values in an array arr[]

here, the address of a value 7 is 2293432. Then the address of *ptr is 2293436

coming to printf();

printf("%d %d\n", *(2293420 + 1), *(2293436 -1));

printf("%d %d\n", *(2293424), *(2293432));

printf("%d %d\n", 2, 7);

thus 2 7

✉ Answer

2. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
        int a[][3] = {0, 1, 2, 3, 4, 5};
        int (*ptr)[3] = a;
        printf("%d %d ", (*ptr)[0], (*ptr)[1]);
        ++ptr;
        printf("%d %d\n", (*ptr)[0], (*ptr)[1]);
        return 0;
}
```

A. 0 1 3 4

B. 0 1 0 1

C. 0 1 2 3

D. 0 1 1 2

x

Option: A

Explanation

Here, *ptr[3] is a pointer array which holds the address of first element in an array a[][3]. Now the address of a[][3] and *ptr[3] are same, which means any changes made to one of the variable will affect other variable.

now *ptr[3] looks like this *ptr[3] = {0, 1, 2}, thus first printf outputted 0 1

In the very next line we have ++ptr;, which pre-increment the address of ptr, i.e) let us consider the address of ptr is 2293432 and after pre-increment the address of ptr will be 2293444 and not 2293436 in this case, because we are incrementing array and not a value in an array.

Now the value of ptr looks like *ptr[3] = {3, 4, 5}, thus second printf outputted 3 4

✉ Answer

3. What will be the output of the C program by considering 'b' as a User input?

```
#include<stdio.h>

int main()

{

        char temp;

        char arr[10] = {1, 2, 3, 4, 5, 6, 9, 8};

        temp = (arr + 1)[2];

        printf("%d\n", temp);

        return 0;

}
```

A. 2

B. 3

C. 4

D. 5

x

Option: C

Explanation

Here, temp = (arr + 1)[2];

Let us consider the address of first element in an array arr[10] is 2293416 then temp looks like this temp = (2293416 + 1)[2];

Now temp =(2293420)[2];, which denotes temp = "index value of 2 from the address 2293420(value = 2)";

Now temp = 4;(address = 2293428)

Thus the program outputted 4.

✉ Answer

4. What will be the code to print 5 contains in a[4][1][0]?

```
#include<stdio.h>

int main()

{

        int a[1][2][3] = {0};

        a[0][1][2] = 5;

        printf("%d",*(*(*(a+0)+1)+2));

        return 0;

}
```

A. printf("%d",*(((a+0)+1)+2));

B. printf("%d",*(*(*(a+0)+1)+2));

C. printf("%d",***((a+0)+1)+2);

D. None of the above

x

Option: B

Explanation

Simply, this is a format for naviting to a value using the address of a first element in an array.

✉ Answer

5. What will be the output of the C program?

```
#include<stdio.h>
void fun(char**);
int main()
{
        char *arr[] = { "bat", "cat", "fat", "hat", "mat", "pat" };
        fun(arr);
        return 0;
}
void fun(char **p)
{
        char *t;
        t = (p += sizeof(int))[-1];
        printf("%s\n", t);
}
```

A. mat

B. fat

C. hat

D. cat

x

Option: C

Explanation

fun(arr) returns the address of first element in an array arr Let we start from the function void fun().

*t is a pointer variable which holds t = (p += sizeof(int))[-1];

ie) t = (p = p + sizeof(int)) [-1];

t = (p = p + 4) [-1];

t = (p = address of bat + 4)[-1];

let us consider a address of bat is 2293416,

t = (p = 2293416 + 4)[-1];

t = (p = 2293432)[-1]

t = ("mat")[-1]; // index from "mat"

t = "hat";

thus hat is outputted.

✉ Answer

6. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
        int arr[5][5][5] = {0};
        printf("%d", ( &arr+1 - &arr ));
        return 0;
}
```

A. 0

B. Compilation error

C. 1

D. 4

x

Option: C

Explanation

printf("%d", (&arr+1 - &arr)); let us consider the address of an array arr starts from 2293420

then, printf("%d", (2293420 +1 - 2293420);

printf("%d", 0 + 1);

printf("%d", 1);

Thus 1 is outputted.

✉ Answer

7. What will be the output of the C program, if input is 6?

```
#include<stdio.h>
void fun(int[][3]);
int main(void)
{
        int arr[3][3] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
        fun(arr);
        printf("%d\n", arr[2][1]);
        return 0;
}
void fun(int b[][3])
{
        ++b;
        b[1][1] = 15;
}
```

A. 15

B. 9

C. 8

D. 7

x

Option: A

Explanation

This question from 2braces.com is more tricky.

fun(arr) returns the address of first element in an array to the function void fun();

Let us consider the address of the values in arr[3][3] is 2293420.

when it passes through the function void fun(int b[][3]), its value is pre-incremented ++b

As it is a multi dimensional array ++b will not skip the address next to the last value in an array arr[][3] instead it skip the address next to first part only i.e) now b[][3] array starts with the address 2293432 (i.e) starts from the value 4 but index from 0, Clearly b[0][0] = 4

Now b[1][1] = 15 will affect the value 8 in arr of array.

Thus arr[2][1] outputted 15.

✉ Answer

8. What will be the output of the C program by considering 'b' as a User input?

```
#include<stdio.h>

int main(){

        int rows = 3, colums = 4, i, j, k;

        int a[3][4] = {1, 2, 3, 5, 7};

        i = j = k = 00;

        for(i = 0;i<rows;i++)

        for(j = 0;j<colums;j++)

        if(a[k][j]<k)
```

```
        k = a[i][j];
        printf("%d\n", k);
        return 0;
}
```

A. 00

B. No output

C. 0

D. 7

x

Option: C

Explanation

Initially we set i = 0, j = 0, k = 0. zero be never greater than any integer values in an array a[3][4], thus if condition fails. and 0 is outputted.

✉ Answer

9. What will be the code to print 5 contains in a[4][1][0]?

```
#include<stdio.h>
int main()
{
        int arr[ ]={1.2, 2.4, 3.6, 4.8, 5};
        int j, *ptr = arr;
        for(j = 0;j<5;j++)
        {
        printf("%d ", *arr);
        ++ptr;
        }
```

}

A. 2 2 2 2 2

B. 1 1 1 1 1

C. 1 2 3 4 5

D. None of the above

x

Option: B

Explanation

Initially array arr is assigned to a pointer variable ptr. In the for loop, ptr is incremented and not arr. So the value 1 1 1 1 1 will be printed. as we use integer type decimal values are all exempted.

✉ Answer

10. What will be the output of the C program?

```
#include<stdio.h>

int main()

{

        int arr[5][5][5] = {0};

        int *b = arr;

        int *c = arr + 1;

        printf("%d", c - b);

        return 0;

}
```

A. 0

B. Runtime Error

C. 25

D. Some address

x

Option: C

Explanation

Clearly array arr[5][5][5] can hold upto 25 integer values.

let us consider the address of first element in an array arr[5][5][5] is 2292932

Now *b = 2292932; *c = arr + 1; i.e) *c contains the address which is located next to the last value address in an arr[5][5][5], which is the address location next to that 25th value in an array arr[5][5][5].

Now *c = 2293032;

here, printf("%d", c-b);

printf("%d", 2292932 - 2293032);

printf("%d", 100); this is not yet over

printf("%d", 100/ sizeof(int)); as it is an integer type values we have to divide it by sizeof(int) to display value not the address.

printf("%d", 25);

thus 25.

✉ Answer

11. What will be the output of the C program?

```
#include<stdio.h>

int main()

{

        int i = 0;

        printf("Hello");

        char s[4] = {'\b', '\t', '\r', '\n'};

        for(i = 0;i<4;i++){
```

```
        printf("%c", s[i]);
}
return 0;
}
```

A. Hello

B. Compilation error

C. Hell

D. None of the above

x

Option: C

Explanation

Hello is printed followed by \b\t\r\n.

i.e) Hello\b\t\r\n.

i.e) Hell\t\r\n.

i.e) Hell \r\n.

i.e) Hell\n.

i.e) Hell is Outputted.

✉ Answer

12. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
        static int a[ ] = {0, 1, 2, 3, 4};
        int *p[ ] = {a, a + 1, a + 2, a + 3, a + 4};
```

```
        int **ptr =  p;
        ++*ptr;
        printf("%d  %d  %d", ptr - p, *ptr - a, **ptr);
        return 0;
}
```

A. 0 1 1

B. 0 0 1

C. 0 1 2

D. 1 1 2

x

Option: A

Explanation

*p[] is a pointer array variable which holds the all 5 addressess of a value in static integer array a[].

Our assumption

Address of 0 in a[] array is 4210692

Then a value of a in *p[] is 4210692 i.e) address of 0 in a[]

Now, the address of a in *p[] array is 2293416.

Our program explanation

**ptr = p;

**ptr = 0;

how **ptr =0?

**ptr = p;

**ptr= address of first element in p[];

We know that **ptr == *(*ptr)

then, *(*ptr) == *(*(address of first element in p[]))

((address of first element in p[])) == *(value of first element in an array p[])

*(value of first element in an array p[]) == *(address of first element in an array a[])

*(address of first element in an array a[]) == value of first element in an array a[];

that is 0

What happens in ++*ptr;

we know that ++(*ptr) == ++(*(address of first element in p[]))

++(*(address of first element in p[])) == ++(value of first element in an array p[])

++(value of first element in an array p[]) == ++(address of first element in a[] which is address of 0)

++(address of first element in a[] which is address of 0) == address of second element in a[] which is address of 1

What happens in printf();

printf ("%d %d %d", ptr-p, *ptr-a, **ptr);

printf ("%d %d %d",2293416-2293416, *ptr-a, **ptr);

printf("%d %d %d", 0/(sizeof (int)), 4210696-4210692, **ptr);

printf("%d %d %d", 0, 4/(sizeof (int)), **ptr);

printf("%d %d %d", 0, 4/4, **ptr);

printf("%d %d %d", 0, 1, **ptr);

printf("%d %d %d", 0, 1, 1);

Thus 0 1 1 is outputted.

✉ Answer

13. What will be the output of the C program?

```
#include<stdio.h>

int main()

{

        int i = 0;
```

```
	char s[4] = {'\0', '\0', '\0', '\0'};
	for(i = 0;i<4;i++)
	{
		printf("%c", s[i]);
	}
	return 0;
}
```

A. \0 \0 \0

B. \0 \0 \0 \0

C. No output

D. None of the above

x

Option: C

Explanation

\0 = NULL. Thus compiler prints nothing.

✉ Answer

14. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
	char s[] = {'a', 'b', 'c', '\n', 'c', '\0'};
	char *p, *str, *str1;
	p = &s[3];
	str = p;
```

```
        str1 = s;
        printf("%d", ++*p + ++*str1-32);
        return 0;
}
```

A. 76

B. 77

C. 78

D. 79

x

Option: B

Explanation

p = &s[3].

i.e) p = address of '\n';

str = p;

i.e) str = address of p;

str1 = s;

str1 = address of 'a';

printf ("%d", ++*p + ++*str1 - 32);

i.e) printf("%d", ++\n + a -32);

i.e) printf("%d", 12 + 97 -32);

i.e) printf("%d", 12 + 65);

i.e) printf("%d", 77);

Thus 77 is outputted.

✉ Answer

15. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
        int i = 0;
        printf("Hello");
        char s[4] = {'\b', '\r', '\t', '\n'};
        for(i = 0;i<4;i++)
        {
                printf("%c", s[i]);
        }
        return 0;
}
```

A. Hello

B. Hell

C. No output

D. Compilation error

x

Option: C

Explanation

Hello is printed followed by \b\r\t\n.

i.e) Hello\b\r\t\n.

i.e) Hell\r\t\n.

i.e) \t\n.

i.e) \n.

i.e) is Outputted.ie(8 space is outputted)

✉ Answer

16. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
        int arr[2] = {1, 2, 3, 4, 5};
        printf("%d", arr[3]);
        return 0;
}
```

A. 3

B. 4

C. Some Garbage value

D. Compilation error

x

Option: C

Explanation

Here the size of an array is 2, but the value inside array is exceed 2. Thus it prints garbage value for index more than 1

✉ Answer

17. What will be the output of the C program?

```
#include<stdio.h>
int main()
```

```
{
        int a, b, c;
        int arr[5] = {1, 2, 3, 25, 7};
        a = ++arr[1];
        b = arr[1]++;
        c = arr[a++];
        printf("%d--%d--%d", a, b, c);
        return 0;
}
```

A. 4--3--25

B. 3--3--25

C. 4--4--25

D. 3--4--25

x

Option: A

Explanation

here, a = ++arr[1];

i.e) a = 3 //arr[2];

b = arr[1]++;

i.e) b = 3 //arr[2];

c = arr[a++];

i.e) c = 25 //arr[4];

It must be noted that a value of a is increment ie) a = 4;

printf("%d--%d--%d",a, b, c);

printf("%d--%d--%d",4, 3, 25);

Thus 4--3--25 is outputted.

✉ Answer

18. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
        int arr[5] = {1, 3, 5, 7, 11};
        int *ptr, *ptr1;
        ptr = &arr;
        ptr1 = *ptr + 3;
        printf("%d--%d", *ptr, ptr1);
}
```

A. 1--11

B. 1-7

C. 1--4

D. 1--some address

x

Option: C

Explanation

Here, ptr = &arr;

ptr = address of a first value in an array arr;

ptr1 = *(address of a first value in an array arr) + 3;

i.e) ptr1 = value of a first element in an array arr + 3;

i.e) ptr1 = 1 + 3;

i.e) ptr1 = 4;

printf("%d--%d", *ptr, ptr1);

printf("%d--%d", 1, 4);

Thus 1--4 is outputted.

✉ Answer

19. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
        int arr[5] = { 1, 3, 5, 7, 11 };
        int *ptr;
        ptr = &arr;
        printf("%d", *ptr + 1);
}
```

A. 1

B. 2

C. 3

D. Runtime error

x

Option: B

Explanation

Here ptr = &arr;

ptr = address of a first value in an array arr;

*ptr = value of a first element in an array arr;

printf("%d", *ptr + 1);

printf("%d", 1 + 1);

Thus 2 is outputted.

✉ Answer

20. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
        static char *arr[ ]  = {"bike", "bus", "car", "van"};
        char **ptr[ ] = {arr+3, arr+2, arr+1, arr};
        char ***p;
        p = ptr;
        **++p;
        printf("%s",*--*++p + 2);
}
```

A. Nothing prints

B. ke

C. ike

D. Compilation error

x

Option: C

Explanation

here, p = ptr;

p = address of first element in an array **ptr = [];

**++p;

i.e) **++(address of first element in an array **ptr = []);

i.e) **(address of second element in an array **ptr = [])

i.e) *(value of second element in an array **ptr[])

the above line is similar to the following line

*(address of car);

i.e) car; // final statement

Now, coming to printf

printf("%s",*--*++p + 2);

first let us examine the value *--*++p

--++p;

--++(address of second element in an array **ptr = [])

--(address of third element in an array **ptr = [])

*--(value of third element in an array **ptr[])

the above line is similar to the following line

*--(address of bus)

*(address of bike)

thus *--*++p = "bike"

Now, printf("%s","bike" + 2);

Thus ke is outputted.

✉ Answer

21. What will be the output of the C program?

#include<stdio.h>

```
#define arr[5] {1, 2, 3, 4, 5}
int main()
{
        printf("%d", arr[1]);
        return 0;
}
```

A. 1

B. 2

C. Compilation error

D. Runtime error

x

Option: C

Explanation

array can't be declared in #define preprocessor.

✉ Answer

22. What will be the output of the C program?

```
#include<stdio.h>
#define arr "abcd"
int main()
{
        printf("%c", arr[2]);
        return 0;
}
```

A. c

B. b

C. Compilation error

D. Runtime error

x

Option: C

Explanation

String can be declared in #define preprocessor.

✉ Answer

23. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
        int arr[1] = {2};
        printf("%d", 0[arr]);
        return 0;
}
```

A. Compilation error

B. Some Garbage value

C. 2

D. 0

x

Option: C

Explanation

Watch clearly, arr[1] = {2}; is similar to

arr[1] = {2, '\0'};

Thus 0[arr] outputted 2

✉ Answer

24. What will be the output of the C program?

```
#include<stdio.h>
void array(int **p);
int main()
{
        int arr[2][3] = {{3, 6, 9 }, {12, 15, 18}};
        int *ptr;
        ptr = &arr;
        array(&ptr);
        return 0;
}
void array(int **p)
{
        printf("%d", **p);
}
```

A. address of first element in array

B. 3

C. address of ptr

D. Runtime error

x

Option: B

Explanation

Here ptr = &arr.

i.e) ptr = address of first element in an array arr[2][3];

array(&ptr);

i.e) array(address of ptr);

Examine void array() funtion

void array(int **p)

i.e) void array(**(address of ptr))

i.e) void array(*(address of first element in an array arr[2][3]))

i.e) void array(value of first element in an array arr[2][3]);

i.e) void array(3)

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

i.e) printf("%d", 3);

Thus 3 is outputted.

✉ Answer

25. What will be the output of the C program?

```
#include<stdio.h>
int main()
{
        int arr[3], i = 0;
        while(i < 3)
        {
                arr[i] = ++i;
        }
```

```
        for(i=0; i<3; i++)
        {
                printf("%d--", arr[i]);
        }
return 0;
}
```

A. Compilation error

B. 1--2--3--

C. Garbage value--1--2--

D. None of the above

x

Option: C

Explanation

Simply arr[0] is left while filling the numbers in array using while loop.

Thus arr[0] = garbage value;

arr[1] = 1;

arr[2] = 2;

Thus outputted Garbage value--1--2--.

✉ Answer