1. What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    int a[5] = {5, 1, 15, 20, 25};
    int i, j, m;
    i = ++a[1];
    j = a[1]++;
    m = a[i++];
    printf("%d, %d, %d", i, j, m);
    return 0;
}
```

A. 2, 1, 15

B. 1, 2, 5

C. 3, 2, 15

D. 2, 3, 20

Answer: Option **C**

Explanation:

Step 1: `int a[5] = {5, 1, 15, 20, 25};` The variable arr is declared as an integer array with a size of 5 and it is initialized to
`a[0] = 5`, `a[1] = 1`, `a[2] = 15`, `a[3] = 20`, `a[4] = 25` .
Step 2: `int i, j, m;` The variable i,j,m are declared as an integer type.
Step 3: `i = ++a[1];` becomes `i = ++1;` Hence `i = 2` and `a[1] = 2`
Step 4: `j = a[1]++;` becomes `j = 2++;` Hence `j = 2` and `a[1] = 3`.
Step 5: `m = a[i++];` becomes `m = a[2];` Hence `m = 15` and `i` is incremented by 1(i++ means 2++ so i=3)
Step 6: `printf("%d, %d, %d", i, j, m);` It prints the value of the variables `i, j, m`

Hence the output of the program is 3, 2, 15

View Answer Discuss in Forum Workspace Report

---

2. What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    static int a[2][2] = {1, 2, 3, 4};
    int i, j;
    static int *p[] = {(int*)a, (int*)a+1, (int*)a+2};
    for(i=0; i<2; i++)
    {
        for(j=0; j<2; j++)
        {
            printf("%d, %d, %d, %d\n", *(*(p+i)+j), *(*(j+p)+i),
                                       *(*(i+p)+j), *(*(p+j)+i));
        }
    }
```

```
    return 0;
}
```

A. 1, 1, 1, 1
2, 3, 2, 3
3, 2, 3, 2
4, 4, 4, 4

B. 1, 2, 1, 2
2, 3, 2, 3
3, 4, 3, 4
4, 2, 4, 2

C. 1, 1, 1, 1
2, 2, 2, 2
2, 2, 2, 2
3, 3, 3, 3

D. 1, 2, 3, 4
2, 3, 4, 1
3, 4, 1, 2
4, 1, 2, 3

Answer: Option **C**
Explanation:
No answer description available for this question. Let us discuss.
View Answer Discuss in Forum Workspace Report

---

3. What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    void fun(int, int[]);
    int arr[] = {1, 2, 3, 4};
    int i;
    fun(4, arr);
    for(i=0; i<4; i++)
        printf("%d,", arr[i]);
    return 0;
}
void fun(int n, int arr[])
{
    int *p=0;
    int i=0;
    while(i++ < n)
        p = &arr[i];
    *p=0;
}
```

A. 2, 3, 4, 5

B. 1, 2, 3, 4

C. 0, 1, 2, 3

D. 3, 2, 1 0

Answer: Option **B**
Explanation:
Step 1: `void fun(int, int[]);` This prototype tells the compiler that the function `fun()` accepts one integer value and one array as an arguments and does not return anything.
Step 2: `int arr[] = {1, 2, 3, 4};` The variable a is declared as an integer array and it is initialized to
`a[0] = 1, a[1] = 2, a[2] = 3, a[3] = 4`
Step 3: `int i;` The variable `i` is declared as an integer type.
Step 4: `fun(4, arr);` This function does not affect the output of the program. Let's skip this function.
Step 5: `for(i=0; i<4; i++) { printf("%d,", arr[i]); }` The `for` loop runs untill the variable `i` is less than '4' and it prints the each value of array `a`.

Hence the output of the program is 1,2,3,4

View Answer Discuss in Forum Workspace Report

---

4. What will be the output of the program ?

```
#include<stdio.h>
void fun(int **p);

int main()
{
    int a[3][4] = {1, 2, 3, 4, 4, 3, 2, 8, 7, 8, 9, 0};
    int *ptr;
    ptr = &a[0][0];
    fun(&ptr);
    return 0;
}
void fun(int **p)
{
    printf("%d\n", **p);
}
```

A. 1

B. 2

C. 3

D. 4

Answer: Option **A**
Explanation:
Step 1: `int a[3][4] = {1, 2, 3, 4, 4, 3, 2, 8, 7, 8, 9, 0};` The variable `a` is declared as an multidimensional integer array with size of 3 rows 4 columns.
Step 2: `int *ptr;` The `*ptr` is a integer pointer variable.
Step 3: `ptr = &a[0][0];` Here we are assigning the base address of the array `a` to the pointer variable `*ptr`.

Step 4: `fun(&ptr);` Now, the `&ptr` contains the base address of array `a`.
Step 4: Inside the function `fun(&ptr);` The `printf("%d\n", **p);` prints the value '1'. because the `*p` contains the base address or the first element memory address of the array `a` (ie. a[0])
`**p` contains the value of *p memory location (ie. a[0]=1).

Hence the output of the program is '1'

View Answer Discuss in Forum Workspace Report

---

5. What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    static int arr[] = {0, 1, 2, 3, 4};
    int *p[] = {arr, arr+1, arr+2, arr+3, arr+4};
    int **ptr=p;
    ptr++;
    printf("%d, %d, %d\n", ptr-p, *ptr-arr, **ptr);
    *ptr++;
    printf("%d, %d, %d\n", ptr-p, *ptr-arr, **ptr);
    *++ptr;
    printf("%d, %d, %d\n", ptr-p, *ptr-arr, **ptr);
    ++*ptr;
    printf("%d, %d, %d\n", ptr-p, *ptr-arr, **ptr);
    return 0;
}
```

A.
0, 0, 0
1, 1, 1
2, 2, 2
3, 3, 3

B.
1, 1, 2
2, 2, 3
3, 3, 4
4, 4, 1

C.
1, 1, 1
2, 2, 2
3, 3, 3
3, 4, 4

D.
0, 1, 2
1, 2, 3
2, 3, 4
3, 4, 5

Answer: Option **C**

6. What will be the output of the program if the array begins at 65472 and each integer occupies 2 bytes?

```
#include<stdio.h>

int main()
{
    int a[3][4] = {1, 2, 3, 4, 4, 3, 2, 1, 7, 8, 9, 0};
    printf("%u, %u\n", a+1, &a+1);
    return 0;
}
```

A. 65474, 65476

B. 65480, 65496

C. 65480, 65488

D. 65474, 65488

Answer: Option **B**
Explanation:
Step 1: `int a[3][4] = {1, 2, 3, 4, 4, 3, 2, 1, 7, 8, 9, 0};` The array a[3][4] is declared as an integer array having the 3 rows and 4 colums dimensions.
Step 2: `printf("%u, %u\n", a+1, &a+1);`

The base address(also the address of the first element) of array is 65472.

For a two-dimensional array like `a` reference to array has type "pointer to array of 4 ints".
Therefore, `a+1` is pointing to the memory location of first element of the second row in array `a`.
Hence 65472 + (4 ints * 2 bytes) = 65480
Then, `&a` has type "pointer to array of 3 arrays of 4 ints", totally 12 ints. Therefore, `&a+1` denotes "12 ints * 2 bytes * 1 = 24 bytes".
Hence, begining address 65472 + 24 = 65496. So, `&a+1` = 65496

Hence the output of the program is 65480, 65496

View Answer Discuss in Forum Workspace Report

---

7. What will be the output of the program in Turb C (under DOS)?

```
#include<stdio.h>

int main()
{
    int arr[5], i=0;
    while(i<5)
        arr[i]=++i;

    for(i=0; i<5; i++)
        printf("%d, ", arr[i]);

    return 0;
}
```

A. 1, 2, 3, 4, 5,

B. Garbage value, 1, 2, 3, 4,

C. 0, 1, 2, 3, 4,

D. 2, 3, 4, 5, 6,

Answer: Option **B**
Explanation:

Since C is a compiler dependent language, it may give different outputs at different platforms. We have given the TurboC Compiler (Windows) output.

Please try the above programs in Windows (Turbo-C Compiler) and Linux (GCC Compiler), you will understand the difference better.

View Answer Discuss in Forum Workspace Report

---

8. What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    int arr[1]={10};
    printf("%d\n", 0[arr]);
    return 0;
}
```

A. 1

B. 10

C. 0

D. 6

Answer: Option **B**
Explanation:
Step 1: `int arr[1]={10};` The variable `arr[1]` is declared as an integer array with size '2' and it's first element is initialized to value '10'(means `arr[0]=10`)
Step 2: `printf("%d\n", 0[arr]);` It prints the first element value of the variable `arr`.

Hence the output of the program is 10.

View Answer Discuss in Forum Workspace Report

---

9. What will be the output of the program if the array begins at address 65486?

```
#include<stdio.h>

int main()
{
    int arr[] = {12, 14, 15, 23, 45};
    printf("%u, %u\n", arr, &arr);
    return 0;
```

```
}
```

A. 65486, 65488

B. 65486, 65486

C. 65486, 65490

D. 65486, 65487

Answer: Option **B**
Explanation:
Step 1: `int arr[] = {12, 14, 15, 23, 45};` The variable `arr` is declared as an integer array and initialized.
Step 2: `printf("%u, %u\n", arr, &arr);` Here,

The base address of the array is 65486.

=> `arr, &arr` is pointing to the base address of the array `arr`.

Hence the output of the program is 65486, 65486

View Answer Discuss in Forum Workspace Report

---

10. What will be the output of the program ?

```
#include<stdio.h>

int main()
{
    float arr[] = {12.4, 2.3, 4.5, 6.7};
    printf("%d\n", sizeof(arr)/sizeof(arr[0]));
    return 0;
}
```

A. 5

B. 4

C. 6

D. 7

Answer: Option **B**
Explanation:

The sizeof function return the given variable. Example: float a=10; sizeof(a) is 4 bytes

Step 1: `float arr[] = {12.4, 2.3, 4.5, 6.7};` The variable `arr` is declared as an floating point array and it is initialized with the values.
Step 2: `printf("%d\n", sizeof(arr)/sizeof(arr[0]));`
The variable `arr` has 4 elements. The size of the float variable is 4 bytes.

Hence 4 elements x 4 bytes = 16 bytes

`sizeof(arr[0])` is 4 bytes
Hence `16/4` is `4` bytes

Hence the output of the program is '4'.

11. What will be the output of the program if the array begins 1200 in memory?

```
#include<stdio.h>

int main()
{
    int arr[]={2, 3, 4, 1, 6};
    printf("%u, %u, %u\n", arr, &arr[0], &arr);
    return 0;
}
```

A. 1200, 1202, 1204

B. 1200, 1200, 1200

C. 1200, 1204, 1208

D. 1200, 1202, 1200

Answer: Option **B**
Explanation:
Step 1: `int arr[]={2, 3, 4, 1, 6};` The variable `arr` is declared as an integer array and initialized.
Step 2: `printf("%u, %u, %u\n", arr, &arr[0], &arr);` Here,

The base address of the array is 1200.

=> `arr, &arr` is pointing to the base address of the array `arr`.
=> `&arr[0]` is pointing to the address of the first element array `arr`. (ie. base address)

Hence the output of the program is 1200, 1200, 1200

1. Which of the following is correct way to define the function `fun()` in the below program?

```
#include<stdio.h>

int main()
{
    int a[3][4];
    fun(a);
    return 0;
}
```

A.

```
void fun(int p[][4])
{
}
```

B.
```
void fun(int *p[4])
{
}
```

C.
```
void fun(int *p[][4])
{
}
```

D.
```
void fun(int *p[3][4])
{
}
```

Answer: Option **A**
Explanation:
`void fun(int p[][4]){ }` is the correct way to write the function `fun()`. while the others are considered only the function `fun()` is called by using `call by reference`.
View Answer Discuss in Forum Workspace Report

---

2. Which of the following statements mentioning the name of the array begins DOES NOT yield the base address?

   1: When array name is used with the `sizeof` operator.

   2: When array name is operand of the & operator.

   3: When array name is passed to `scanf()` function.

   4: When array name is passed to `printf()` function.

   A. A

   B. A, B

   C. B

   D. B, D

   Answer: Option **B**
   Explanation:
   The statement 1 and 2 does not yield the base address of the array. While the `scanf()` and `printf()` yields the base address of the array.
   View Answer Discuss in Forum Workspace Report

---

3. Which of the following statements are correct about the program below?

```
#include<stdio.h>

int main()
{
    int size, i;
    scanf("%d", &size);
```

```
    int arr[size];
    for(i=1; i<=size; i++)
    {
        scanf("%d", arr[i]);
        printf("%d", arr[i]);
    }
    return 0;
}
```

A. The code is erroneous since the subscript for array used in `for` loop is in the range 1 to `size`.

B. The code is erroneous since the values of array are getting scanned through the loop.

C. The code is erroneous since the statement declaring array is invalid.

D. The code is correct and runs successfully.

Answer: Option **C**
Explanation:
The statement `int arr[size];` produces an error, because we cannot initialize the size of array dynamically. Constant expression is required here.
Example: `int arr[10];`
One more point is there, that is, usually declaration is not allowed after calling any function in a current block of code. In the given program the declaration `int arr[10];` is placed after a function call `scanf()`.
View Answer Discuss in Forum Workspace Report

---

4. Which of the following statements are correct about 6 used in the program?

```
int num[6];
num[6]=21;
```

A. In the first statement 6 specifies a particular element, whereas in the second statement it specifies a type.

B. In the first statement 6 specifies a array size, whereas in the second statement it specifies a particular element of array.

C. In the first statement 6 specifies a particular element, whereas in the second statement it specifies a array size.

D. In both the statement 6 specifies array size.

Answer: Option **B**
Explanation:
The statement 'B' is correct, because `int num[6];` specifies the size of array and `num[6]=21;` designates the particular element(7^{th} element) of the array.
View Answer Discuss in Forum Workspace Report

---

5. Which of the following statements are correct about an array?
   1: The array `int num[26];` can store 26 elements.
   2: The expression `num[1]` designates the very first element in the array.

3: It is necessary to initialize the array at the time of declaration.

4: The declaration `num[SIZE]` is allowed if SIZE is a macro.

A. 1

B. 1,4

C. 2,3

D. 2,4

Answer: Option **B**

Explanation:

1. The array `int num[26];` can store 26 elements. This statement is true.
2. The expression `num[1]` designates the very first element in the array. This statement is false, because it designates the second element of the array.

3. It is necessary to initialize the array at the time of declaration. This statement is false.

4. The declaration `num[SIZE]` is allowed if SIZE is a macro. This statement is true, because the MACRO just replaces the symbol SIZE with given value.

Hence the statements '1' and '4' are correct statements.