

PROGRAMMING IN JAVA

Assignment 0

TYPE OF QUESTION: MCQ

Number of questions: 10

Total marks: $10 \times 1 = 10$

QUESTION 1:

What will be the output of the following code snippet?

```
#include <stdio.h>
void solve() {
    int a = 3;
    int res = a++ + ++a + a++ + ++a;
    printf("%d", res);
}
int main() {
    solve();
    return 0;
}
```

- a. 12
- b. 24
- c. 20**
- d. 18

Correct Answer: c. 20

Detailed Solution:

In the given code snippet, the `solve()` function is called from the `main()` function. Inside the `solve()` function: `int a = 3;` initializes the variable `a` with the value 3. `int res = a++ + ++a + a++ + ++a;` performs a series of arithmetic operations and assigns the result to the variable `res`. `a++` is a post-increment operation, which means the value of `a` is used and then incremented by 1. So, initially, `a++` evaluates to 3, and `a` becomes 4. `++a` is a pre-increment operation, which means the value of `a` is incremented by 1 and then used. So, after the previous operation, `++a` evaluates to 5, and `a` becomes 5. `a++` is another post-increment operation. At this point, `a` is 5, and the expression evaluates to 5. Then, `a` becomes 6. `++a` is another pre-increment operation. After the previous operation, `++a` evaluates to 7, and `a` becomes 7. Therefore, the overall expression becomes `3 + 5 + 5 + 7`, which results in 20 and is stored in the variable `res`. `printf("%d", res);` prints the value of `res`, which is 20.



QUESTION 2:

What will be the value of `x` in the following code snippet?

```
#include <stdio.h>
void solve() {
    int x = printf("Hello");
    printf(" %d", x);
}
int main() {
    solve();
    return 0;
}
```

- a. 10
- b. 5**
- c. 1
- d. 0

Correct Answer: b. 5

Detailed Solution:

`printf` is a library function defined under `stdio.h` header file that returns the count of characters printed in `STDOUT`. In this case it prints *"Hello"*, which is 5 characters, thus the value returned to `x` is 5.



QUESTION 3:

Pick the best option for the following statement:

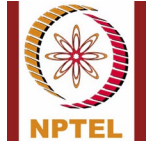
```
int (*p)[5];
```

- a. The statement will create an array `p` of pointers.
- b. The statement has an error.
- c. The statement will create an array `p` of integers.
- d. None of the above

Correct Answer: a. The statement will create an array `p` of pointers.

Detailed Solution:

The statement does not have an error and will compile normally. `"int (*p)[5];"` is same as `"int* p[5];"` which is used to declare an array of pointers.



QUESTION 4:

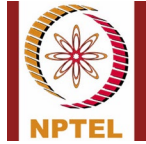
The size of a union is determined by:

- a. The sum of all the members' sizes
- b. The biggest member of the union
- c. The last member of the union
- d. The first member of the union

Correct Answer: b. The biggest member of the union

Detailed Solution:

In a union, all members share the same memory space, and the size of the union is determined by the size of its largest member. This is because the memory allocated for a union needs to be large enough to accommodate the largest member so that it can hold any of the union's members. When calculating the size of a union, it is sufficient to consider only the largest member. The sizes of other members do not contribute to the overall size of the union since they share the same memory space.



QUESTION 5:

What will be the output of the following code snippet?

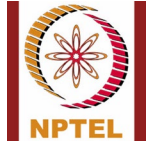
```
#include <stdio.h>
int main()
{
    int *ptr, a = 10;
    ptr = &a;
    *ptr += 1;
    printf("%d,%d", *ptr, a);
}
```

- a. 10, 10
- b. 10, 11
- c. 11, 10
- d. 11, 11**

Correct Answer: d. 11, 11

Detailed Solution:

Since `ptr` is just a pointer to `a`, when the value of `a` changes, so does the value in the pointer. Thus `*ptr += 1;` will increment the value of `a` as well.



QUESTION 6:

What will be the output of the following code snippet?

```
#include <stdio.h>
// Assume base address of "Test_Quiz" to be 1000
int main()
{
    printf(5 + "Test_Quiz");
    return 0;
}
```

- a. Test
- b. Quiz**
- c. 1005
- d. Compile-time error

Correct Answer: b. Quiz

Detailed Solution:

`printf` is a library function defined under `stdio.h` header file. The compiler adds 5 to the base address of the string through the expression `5 + \"Test_Quiz\"`. Then the string `\"Quiz\"` gets passed to the standard library function as an argument. Then the string is printed normally without error.



QUESTION 7:

Which of the following is not a storage class specifier in C?

- a. auto
- b. register
- c. static
- d. volatile

Correct Answer: d. volatile

Detailed Solution:

`volatile` is not a storage class specifier in C. `volatile` and `const` are type qualifiers.



QUESTION 8:

In C, static storage class cannot be used with:

- a. Global variable
- b. Function parameter**
- c. Function name
- d. Local variable

Correct Answer: b. Function parameter

Detailed Solution:

Declaring a global variable as static limits its scope to the same file in which it is defined. A static function is only accessible to the same file in which it is defined. A local variable declared as static preserves the value of the variable between the function calls.



QUESTION 9:

What will be the output of the following code snippet?

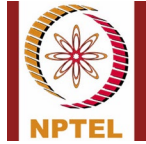
```
#include<stdio.h>
int main()
{
    char str[20] = "Test_Quiz";
    printf ("%d", sizeof(str));
    return 0;
}
```

- a. 9
- b. 10
- c. Garbage value
- d. 20**

Correct Answer: d. 20

Detailed Solution:

The `sizeof()` operator would return size of array. To get size of string stored in array, we need to use `strlen()`. The following program prints 20 which is the size of the array.



QUESTION 10:

Point out the error, if any, in the for loop:

```
#include<stdio.h>
int main()
{
    int i = 1;
    for (;;)
    {
        printf("%d", i++);
        if (i > 10)
            break;
    }
}
```

- a. ERROR: The condition part in the `for` loop is required.
- b. ERROR: The two semicolons should be dropped.
- c. ERROR: The `for` loop should be replaced by a `while` loop.
- d. No error.

Correct Answer: d. No error.

Detailed Solution:

All components of a for loop is optional and there is no syntax error in the code.

QUESTION 11:

What is the following set of statements equivalent to?

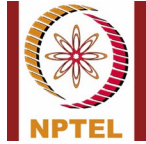
```
if(x==1)
{
    x=0;
}
else if(x==0)
{
    x=1;
}
```

- a. `x = 1 + x;`
- b. `x = 1 - x;`
- c. `x = x - 1;`
- d. `x = 1 % x;`

Correct Answer: b. `x = 1 - x;`

Detailed Solution:

The above code inverts the value of `x`, i.e., performs a NOT operation. Using `x = 1 - x;` we can get the same effect in just a single line. If `x` is 1 then `1-x` will set the value of `x` as 0 and vice versa.



QUESTION 12:

What is defined by the statements given below?

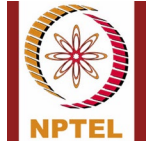
```
struct node
{
    int i;
    float j;
};
struct node *s[10];
```

- a. An array, each element of which is pointer to a structure of type `node`.
- b. A structure of 2 fields, each field being a pointer to an array of 10 elements.
- c. A structure of 3 fields: an integer, a float, and an array of 10 elements.
- d. An array, each element of which is a structure of type `node`.

Correct Answer: a. An array, each element of which is pointer to a structure of type `node`

Detailed Solution:

`struct node *s[10];` defines a pointer array of 10 elements of type `struct node*`.



QUESTION 13:

What needs to be called in order to exchange the values of two variables `x` and `y` in the following code?

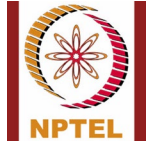
```
void swap (int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
```

- a. call `swap(x, y)`
- b. call `swap(&x, &y)`
- c. call `swap(*x, *y)`
- d. `swap(x, y)` cannot be used to swap the variables

Correct Answer: d. `swap(x, y)` cannot be used as the parameters are passed by value

Detailed Solution:

`swap()` function will only swap the values of local variables `a` and `b`, however, there will be no effect on `x` and `y`, i.e. they will remain unchanged.



QUESTION 14:

What will be the output of the program if the array begins 2200 in memory?

```
#include<stdio.h>

int main()
{
    int arr[]={12,13, 14, 1, 6};
    printf("%u, %u, %u", arr, &arr[0], &arr);
    return 0;
}
```

- a. 2300, 2200, 2500
- b. 2200, 2200, 2200**
- c. 2300, 2400, 2500
- d. 2200, 2200, 2300

Correct Answer: b. 2200, 2200, 2200

Detailed Solution:

`arr`, `&arr[0]`, `&arr` all refer to base address of 2200.



QUESTION 15:

In C/ C++, if you pass an array as an argument to a function, what actually gets passed?

- a. Value of elements in array
- b. First element of the array
- c. Base address of the array
- d. Address of the last element of array

Correct Answer: c. Base address of the array

Detailed Solution:

When we pass an array as a function argument, the base address of the array will be passed.

*****END*****