

1) The correct order of compilation of a C program is

- i) Compile source code to object code
- ii) Link object code and libraries
- iii) Create binary or executable file
- iv) Process Preprocessor directives

- A) i → ii → iii → iv
- B) iv → i → ii → iii**
- C) i → iv → ii → iii

**Answer:** Preprocessor directives are processed first, followed by compilation, linking and binary file creation.

2) Which of the following data type can be used for storing a floating point constant?

- A) long int
- B) double**
- C) unsigned char

**Answer:** float & double data types can be used for storing floating point constants in C.

3) What is the result of running the following code snippet?

```
float result;  
result = 5/2;  
printf("%.1f",result);
```

- A) 2.0**
- B) 2.5
- C) 0

**Answer:** Division of two integer constants results in an integer in C. So 5/2 becomes 2. As the output is printed in %.1f format, it is printed as 2.0

4) What is the result of running the following code snippet

```
if(0)  
    printf("The Sun rises in the East");  
else  
    printf("The Sun rises in the West");
```

- A) The Sun rises in the East
- B) The Sun rises in the West**
- C) Compilation Error

**Answer:** Non-zero is considered true and zero is false in C. So the statement in *else* is printed.

5) What is the value of variable 'i' after running the following statements?

```
int i = 10;
i = i / 2;
i = i % 2;
A) 0
B) 1
C) 2
```

**Answer:** 10/2 is 5. 5%2 is 1. % is modular division operator.

6) The purpose of the function *int toupper (int c)* as defined in ctype.h is to

- A) Convert the input string stored in variable c to upper case
- B) Convert the input character stored in variable c to upper case
- C) None of the above

**Answer:** toupper converts input character to upper case. Even none of the above is awarded marks as there is ambiguity in answers A, B. The answers were meant to mean "Convert the input character stored in variable c to upper case." The answers give a wrong meaning like "Convert the input **character** 'c' to upper case. Like 'c' to 'C'".

7) The difference between the user defined constants 'Z' and "Z" is

- A) 'Z' is a string constant whereas "Z" is a character constant
- B) 'Z' is a character constant whereas "Z" is a string constant
- C) There is no difference

**Answer:** Character constants are enclosed within single inverted comma or apostrophe '. Whereas, strings in C are enclosed within double inverted commas "".

8) Output of running the following code snippet is

```
int i;
for ( i = 0; i < 3; i++ )
{
    if(i == 2)
    {
        printf("%d\t",i);
    }
}
A) 0    1    2    3
B) 0    1    2
C) 2
```

**Answer:** No explanation required.

9) The conversion character for printing an integer in hexadecimal number using *printf* is

- A) %x
- B) %d
- C) %o

**Answer:** No explanation required.

10) Output of following code snippet

```
float marks[10] = {53,66,36,36,53,23,64,12,53,88};  
float *marksPtr = marks;  
marksPtr = marksPtr + 4;  
printf("%.0f", *marksPtr);
```

- A) 0
- B) 53
- C) 36
- D) Garbage value

**Answer:** marksPtr + 4 points to the 5<sup>th</sup> element in array. Therefore 53.

11) Which of the following is the correct way to allocate space for an array of 15 float variables?

- A) float \*floatArray = (float \*) malloc ( 15 );
- B) float floatArray = (float \*) malloc ( 15 );
- C) float \*floatArray = (float \*) malloc ( 15 \* sizeof(float) );
- D) float \*floatArray = 15 \* malloc (sizeof(float) );

**Answer:** No explanation required. This is syntax of malloc.

12) The following C statement causes an infinite loop (true/false):

**while (1);**

- A) true
- B) false

**Answer:** 1 is true. So *while* gets stuck in an infinite loop.

13) Output of the following code snippet is:

```
int array[] = {1,2,3,4,5};  
printf("%d", array[5]);
```

- A) 0
- B) 4
- C) 5
- D) Garbage value

**Answer:** array[5] refer to 6<sup>th</sup> element in the array. Hence garbage value.

14) Output of following statement is

```
#define NUM 5
int main()
{
    NUM++;
    printf("%d", NUM);
    return 0;
}
```

- A) 5
- B) 6
- C) Compilation error

**Answer:** NUM is a symbolic constant. A constant's value can't be modified.

15) Output of the following statement is

```
int i = 5;
if(i = 5)
{
    puts("Structured programming language");
}
else
{
    puts("Object oriented programming language");
}
```

- A) Structured programming language
- B) Object oriented programming language
- C) Compilation error

**Answer:** i=5 is an assignment, not a comparison. Hence the statement becomes if(5) and hence prints statement in if block.

16) Which of the following is a true statement?

- A) Strings in C are char arrays terminated by '\0'
- B) Strings in C are char arrays terminated by NULL
- C) Strings in C are char arrays terminated by EOF

**Answer:** No explanation required.

17) Which of the following is a NOT a true statement?

- A) The base address of an array is same as the name of the array
- B) The base address of an array is address of the first element in the array
- C) The base address of a static array can be modified

**Answer:** No explanation required.

- 18) Which of the following is true about break statement in C?
- A) break statement exits out of the innermost loop
  - B) break statement exits out of all the outer loops nested around it
  - C) break statement skips one iteration and continues looping

**Answer:**

Consider a loop

```
for(i=0; i<5; i++)
    for(j=0; j<6; j++)
        break;
```

break statement exits out of the inner most loop nested around it.

- 19) Output of the following program is:

```
int main()
{
    int i =0;
    modify(i);
    printf("%d", i);
    return 0;
}
int modify(int z)
{
    return (++z);
}
```

- A) 0
- B) 1

**Answer:** A copy of *i* is sent to the function modify, hence changes to that copy are not reflected in *i*. Call by value concept.

- 20) Point out at least two errors in the following code snippet
- ```
float a;
scanf("Enter number: %f", a )
```

**Answer:**

1. String "Enter number" can't be entered in scanf format string
2. & is missing for a. it should have been scanf("%f", &a );
3. Semicolon missing at the end of the statement