C Programming Interview Questions

A list of 50 top frequently asked C programming interview questions and answers are given below.

1) What is C language?

C is a mid-level and procedural programming language. The Procedural programming language is also known as the structured programming language is a technique in which large programs are broken down into smaller modules, and each module uses structured code. This technique minimizes error and misinterpretation. More details.

---

2) Why is C known as a mother language?

C is known as a mother language because most of the compilers and JVMs are written in C language. Most of the languages which are developed after C language has borrowed heavily from it like C++, Python, Rust, javascript, etc. It introduces new core concepts like arrays, functions, file handling which are used in these languages. More details.

---

3) Why is C called a mid-level programming language?

C is called a mid-level programming language because it binds the low level and high -level programming language. We can use C language as a System programming to develop the operating system as well as an Application programming to generate menu driven customer driven billing system. More details.

---

4) Who is the founder of C language?

Dennis Ritchie. More details.

---

5) When was C language developed?

C language was developed in 1972 at bell laboratories of AT&T. More details.

---

6) What are the features of the C language?

The main features of C language are given below:

- **Simple:** C is a simple language because it follows the structured approach, i.e., a program is broken into parts

- **Portable:** C is highly portable means that once the program is written can be run on any machine with little or no modifications.
- **Mid Level:** C is a mid-level programming language as it combines the low- level language with the features of the high-level language.
- **Structured:** C is a structured language as the C program is broken into parts.
- **Fast Speed:** C language is very fast as it uses a powerful set of data types and operators.
- **Memory Management:** C provides an inbuilt memory function that saves the memory and improves the efficiency of our program.
- **Extensible:** C is an extensible language as it can adopt new features in the future.

More details.

---

7) What is the use of printf() and scanf() functions?

printf(): The printf() function is used to print the integer, character, float and string values on to the screen.

Following are the format specifier:

- **%d**: It is a format specifier used to print an integer value.
- **%s**: It is a format specifier used to print a string.

- **%c**: It is a format specifier used to display a character value.
- **%f**: It is a format specifier used to display a floating point value.

scanf(): The scanf() function is used to take input from the user.

More details.

---

8) What is the difference between the local variable and global variable in C?

Following are the differences between a local variable and global variable:

Basis for comparison	**Local variable**	**Global variable**
Declaration	A variable which is declared inside function or block is known as a local variable.	A variable which is declared outside function or block is known as a global variable.
Scope	The scope of a variable is available within a function in which they are declared.	The scope of a variable is available throughout the program.

Access	Variables can be accessed only by those statements inside a function in which they are declared.	Any statement in the entire program can access variables.
Life	Life of a variable is created when the function block is entered and destroyed on its exit.	Life of a variable exists until the program is executing.
Storage	Variables are stored in a stack unless specified.	The compiler decides the storage location of a variable.

More details.

---

9) What is the use of a static variable in C?

Following are the uses of a static variable:

- A variable which is declared as static is known as a static variable. The static variable retains its value between multiple function calls.
- Static variables are used because the scope of the static variable is available in the entire program. So, we can access a static variable anywhere in the program.
- The static variable is initially initialized to zero. If we update the value of a variable, then the updated value is assigned.

- The static variable is used as a common value which is shared by all the methods.
- The static variable is initialized only once in the memory heap to reduce the memory usage.

More details.

---

10) What is the use of the function in C?

Uses of C function are:

- C functions are used to avoid the rewriting the same code again and again in our program.
- C functions can be called any number of times from any place of our program.
- When a program is divided into functions, then any part of our program can easily be tracked.
- C functions provide the reusability concept, i.e., it breaks the big task into smaller tasks so that it makes the C program more understandable.

More details.

---

11) What is the difference between call by value and call by reference in C?

Following are the differences between a call by value and call by reference are:

	Call by value	Call by reference
Description	When a copy of the value is passed to the function, then the original value is not modified.	When a copy of the value is passed to the function, then the original value is modified.
Memory location	Actual arguments and formal arguments are created in separate memory locations.	Actual arguments and formal arguments are created in the same memory location.
Safety	In this case, actual arguments remain safe as they cannot be modified.	In this case, actual arguments are not reliable, as they are modified.
Arguments	The copies of the actual arguments are passed to the formal arguments.	The addresses of actual arguments are passed to their respective formal arguments.

Example of call by value:

```
1. #include <stdio.h>
2. void change(int,int);
3. int main()
4. {
```

```
5.    int a=10,b=20;
6.    change(a,b); //calling a function by passing the values of variables.
7.    printf("Value of a is: %d",a);
8.    printf("\n");
9.    printf("Value of b is: %d",b);
10.        return 0;
11.    }
12.    void change(int x,int y)
13.    {
14.        x=13;
15.        y=17;
16.    }
```

Output:

```
Value of a is: 10
Value of b is: 20
```

Example of call by reference:

```
1. #include <stdio.h>
2. void change(int*,int*);
3. int main()
```

```
4. {
5.     int a=10,b=20;
6.     change(&a,&b); // calling a function by passing references of variables.
7.     printf("Value of a is: %d",a);
8.     printf("\n");
9.     printf("Value of b is: %d",b);
10.         return 0;
11.     }
12.     void change(int *x,int *y)
13.     {
14.         *x=13;
15.         *y=17;
16.     }
```

Output:

```
Value of a is: 13
Value of b is: 17
```

More details.

12) What is recursion in C?

When a function calls itself, and this process is known as recursion. The function that calls itself is known as a recursive function.

Recursive function comes in two phases:

1. Winding phase
2. Unwinding phase

Winding phase: When the recursive function calls itself, and this phase ends when the condition is reached.

Unwinding phase: Unwinding phase starts when the condition is reached, and the control returns to the original call.

Example of recursion

```
1. #include <stdio.h>
2. int calculate_fact(int);
3. int main()
4. {
5.  int n=5,f;
6.  f=calculate_fact(n); // calling a function
```

```
7.  printf("factorial of a number is %d",f);
8.   return 0;
9. }
10.      int calculate_fact(int a)
11.      {
12.        if(a==1)
13.        {
14.           return 1;
15.        }
16.        else
17.        return a*calculate_fact(a-1); //calling a function recursively.
18.         }
```

Output:

```
factorial of a number is 120
```

More details.

---

13) What is an array in C?

An Array is a group of similar types of elements. It has a contiguous memory location. It makes the code optimized, easy to traverse and easy to sort. The size and type of arrays cannot be changed after its declaration.

Arrays are of two types:

- **One-dimensional array**: One-dimensional array is an array that stores the elements one after the another.

Syntax:

```
1. data_type array_name[size];
```

- **Multidimensional array**: Multidimensional array is an array that contains more than one array.

Syntax:

```
1. data_type array_name[size];
```

Example of an array:

```
1. #include <stdio.h>
2. int main()
3. {
```

```
4.    int arr[5]={1,2,3,4,5}; //an array consists of five integer values.
5.    for(int i=0;i<5;i++)
6.    {
7.        printf("%d ",arr[i]);
8.    }
9.     return 0;
10.       }
```

Output:

```
1 2 3 4 5
```

More details.

---

14) What is a pointer in C?

A pointer is a variable that refers to the address of a value. It makes the code optimized and makes the performance fast. Whenever a variable is declared inside a program, then the system allocates some memory to a variable. The memory contains some address number. The variables that hold this address number is known as the pointer variable.

For example:

```
1. Data_type *p;
```

The above syntax tells that p is a pointer variable that holds the address number of a given data type value.

Example of pointer

```
1. #include <stdio.h>
2. int main()
3. {
4.    int *p; //pointer of type integer.
5.    int a=5;
6.    p=&a;
7.    printf("Address value of 'a' variable is %u",p);
8.     return 0;
9. }
```

Output:

```
Address value of 'a' variable is 428781252
```

More details.

---

15) What is the usage of the pointer in C?

- **Accessing array elements**: Pointers are used in traversing through an array of integers and strings. The string is an array of characters which is terminated by a null character '\0'.
- **Dynamic memory allocation**: Pointers are used in allocation and deallocation of memory during the execution of a program.
- **Call by Reference**: The pointers are used to pass a reference of a variable to other function.
- **Data Structures like a tree, graph, linked list, etc.**: The pointers are used to construct different data structures like tree, graph, linked list, etc.

---

16) What is a NULL pointer in C?

A pointer that doesn't refer to any address of value but NULL is known as a NULL pointer. When we assign a '0' value to a pointer of any type, then it becomes a Null pointer.

More details.

---

17) What is a far pointer in C?

A pointer which can access all the 16 segments (whole residence memory) of RAM is known as far pointer. A far pointer is a 32-bit pointer that obtains information outside the memory in a given section.

18) What is dangling pointer in C?

- If a pointer is pointing any memory location, but meanwhile another pointer deletes the memory occupied by the first pointer while the first pointer still points to that memory location, the first pointer will be known as a dangling pointer. This problem is known as a dangling pointer problem.
- Dangling pointer arises when an object is deleted without modifying the value of the pointer. The pointer points to the deallocated memory.

Let's see this through an example.

```
1. #include<stdio.h>
2. void main()
3. {
4.         int *ptr = malloc(constant value); //allocating a memory space.
5.         free(ptr); //ptr becomes a dangling pointer.
6. }
```

In the above example, initially memory is allocated to the pointer variable ptr, and then the memory is deallocated from the pointer variable. Now, pointer variable, i.e., ptr becomes a dangling pointer.

How to overcome the problem of a dangling pointer

The problem of a dangling pointer can be overcome by assigning a NULL value to the dangling pointer. Let's understand this through an example:

```
1. #include<stdio.h>
2.     void main()
3.     {
4.         int *ptr = malloc(constant value); //allocating a memory space.
5.         free(ptr); //ptr becomes a dangling pointer.
6.         ptr=NULL; //Now, ptr is no longer a dangling pointer.
7.     }
```

In the above example, after deallocating the memory from a pointer variable, ptr is assigned to a NULL value. This means that ptr does not point to any memory location. Therefore, it is no longer a dangling pointer.

---

19) What is pointer to pointer in C?

In case of a pointer to pointer concept, one pointer refers to the address of another pointer. The pointer to pointer is a chain of pointers. Generally, the pointer contains the address of a variable. The pointer to pointer contains the address of a first pointer. Let's understand this concept through an example:

```
1. #include <stdio.h>
2.  int main()
3. {
4.     int a=10;
5.     int *ptr,**pptr; // *ptr is a pointer and **pptr is a double pointer.
6.     ptr=&a;
7.     pptr=&ptr;
8.     printf("value of a is:%d",a);
9.     printf("\n");
10.         printf("value of *ptr is : %d",*ptr);
11.         printf("\n");
12.         printf("value of **pptr is : %d",**pptr);
13.         return 0;
14.     }
```

In the above example, pptr is a double pointer pointing to the address of the ptr variable and ptr points to the address of 'a' variable.

More details.

---

20) What is static memory allocation?

- In case of static memory allocation, memory is allocated at compile time, and memory can't be increased while executing the program. It is used in the array.
- The lifetime of a variable in static memory is the lifetime of a program.
- The static memory is allocated using static keyword.
- The static memory is implemented using stacks or heap.
- The pointer is required to access the variable present in the static memory.
- The static memory is faster than dynamic memory.
- In static memory, more memory space is required to store the variable.

1. For example:
2. **int** a[10];

The above example creates an array of integer type, and the size of an array is fixed, i.e., 10.

More details.

---

21) What is dynamic memory allocation?

- In case of dynamic memory allocation, memory is allocated at runtime and memory can be increased while executing the program. It is used in the linked list.
- The malloc() or calloc() function is required to allocate the memory at the runtime.

- An allocation or deallocation of memory is done at the execution time of a program.
- No dynamic pointers are required to access the memory.
- The dynamic memory is implemented using data segments.
- Less memory space is required to store the variable.

1. For example
2. **int** *p= malloc(**sizeof**(**int**)*10);

The above example allocates the memory at runtime.

More details.

---

22) What functions are used for dynamic memory allocation in C language?

1. malloc()
    - The malloc() function is used to allocate the memory during the execution of the program.
    - It does not initialize the memory but carries the garbage value.
    - It returns a null pointer if it could not be able to allocate the requested space.

    Syntax

1. ptr = (cast-type*) malloc(byte-size) // allocating the memory using malloc() function.

2. calloc()
    - The calloc() is same as malloc() function, but the difference only is that it initializes the memory with zero value.

Syntax

```
1. ptr = (cast-type*)calloc(n, element-size);// allocating the memory using calloc() function.
```

2. realloc()
    - The realloc() function is used to reallocate the memory to the new size.
    - If sufficient space is not available in the memory, then the new block is allocated to accommodate the existing data.

Syntax

```
1. ptr = realloc(ptr, newsize); // updating the memory size using realloc() function.
```

In the above syntax, ptr is allocated to a new size.

2. free():The free() function releases the memory allocated by either calloc() or malloc() function.

Syntax

```
1. free(ptr); // memory is released using free() function.
```

The above syntax releases the memory from a pointer variable ptr.

More details.

---

23) What is the difference between malloc() and calloc()?

	calloc()	malloc()
Description	The malloc() function allocates a single block of requested memory.	The calloc() function allocates multiple blocks of requested memory.
Initialization	It initializes the content of the memory to zero.	It does not initialize the content of memory, so it carries the garbage value.
Number of arguments	It consists of two arguments.	It consists of only one argument.
Return value	It returns a pointer pointing to the allocated memory.	It returns a pointer pointing to the allocated memory.

More details.

24) What is the structure?

- The structure is a user-defined data type that allows storing multiple types of data in a single unit. It occupies the sum of the memory of all members.
- The structure members can be accessed only through structure variables.
- Structure variables accessing the same structure but the memory allocated for each variable will be different.

Syntax of structure

```
1. struct structure_name
2. {
3.   Member_variable1;
4.  Member_variable2
5. .
6. .
7. }[structure variables];
```

Let's see a simple example.

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

```
2. struct student
3. {
4.     char name[10];       // structure members declaration.
5.     int age;
6. }s1;      //structure variable
7. int main()
8. {
9.     printf("Enter the name");
10.          scanf("%s",s1.name);
11.          printf("\n");
12.          printf("Enter the age");
13.          scanf("%d",&s1.age);
14.          printf("\n");
15.          printf("Name and age of a student: %s,%d",s1.name,s1.age);
16.          return 0;
17.      }
```

Output:

```
Enter the name shikha
Enter the age 26
Name and age of a student: shikha,26
```

More details.

---

25) What is a union?

- The union is a user-defined data type that allows storing multiple types of data in a single unit. However, it doesn't occupy the sum of the memory of all members. It holds the memory of the largest member only.
- In union, we can access only one variable at a time as it allocates one common space for all the members of a union.

Syntax of union

```
1. union union_name
2. {
3. Member_variable1;
4. Member_variable2;
5. .
6. .
7. Member_variable n;
8. }[union variables];
```

Let's see a simple example

```
1. #include<stdio.h>
2. union data
3. {
4.     int a;      //union members declaration.
5.     float b;
6.     char ch;
7. };
8. int main()
9. {
10.         union data d;       //union variable.
11.         d.a=3;
12.         d.b=5.6;
13.         d.ch='a';
14.         printf("value of a is %d",d.a);
15.         printf("\n");
16.         printf("value of b is %f",d.b);
17.         printf("\n");
18.         printf("value of ch is %c",d.ch);
19.         return 0;
20.      }
```

Output:

```
value of a is 1085485921
value of b is 5.600022
value of ch is a
```

In the above example, the value of a and b gets corrupted, and only variable ch shows the actual output. This is because all the members of a union share the common memory space. Hence, the variable ch whose value is currently updated.

More details.

---

26) What is an auto keyword in C?

In C, every local variable of a function is known as an automatic (auto) variable. Variables which are declared inside the function block are known as a local variable. The local variables are also known as an auto variable. It is optional to use an auto keyword before the data type of a variable. If no value is stored in the local variable, then it consists of a garbage value.

---

27) What is the purpose of sprintf() function?

The sprintf() stands for "string print." The sprintf() function does not print the output on the console screen. It transfers the data to the buffer. It returns the total number of characters present in the string.

Syntax

```
1. int sprintf ( char * str, const char * format, ... );
```

Let's see a simple example

```
1.  #include<stdio.h>
2. int main()
3. {
4.  char a[20];
5.  int n=sprintf(a,"javaToint");
6.  printf("value of n is %d",n);
7.  return 0;}
```

Output:

```
value of n is 9
```

---

28) Can we compile a program without main() function?

Yes, we can compile, but it can't be executed.

But, if we use #define, we can compile and run a C program without using the main() function. For example:

```
1. #include<stdio.h>
2. #define start main
3. void start() {
4.    printf("Hello");
5. }
```

More details.

---

29) What is a token?

The Token is an identifier. It can be constant, keyword, string literal, etc. A token is the smallest individual unit in a program. C has the following tokens:

1. Identifiers: Identifiers refer to the name of the variables.
2. Keywords: Keywords are the predefined words that are explained by the compiler.
3. Constants: Constants are the fixed values that cannot be changed during the execution of a program.

4. Operators: An operator is a symbol that performs the particular operation.
5. Special characters: All the characters except alphabets and digits are treated as special characters.

---

30) What is command line argument?

The argument passed to the main() function while executing the program is known as command line argument. For example:

```
1. main(int count, char *args[]){
2. //code to  be executed
3. }
```

More details.

---

31) What is the acronym for ANSI?

The ANSI stands for " American National Standard Institute." It is an organization that maintains the broad range of disciplines including photographic film, computer languages, data encoding, mechanical parts, safety and more.

---

32) What is the difference between getch() and getche()?

The **getch()** function reads a single character from the keyboard. It doesn't use any buffer, so entered data will not be displayed on the output screen.

The **getche()** function reads a single character from the keyword, but data is displayed on the output screen. Press Alt+f5 to see the entered character.

Let's see a simple example

```
1. #include<stdio.h>
2. #include<conio.h>
3. int main()
4. {
5.
6.  char ch;
7.  printf("Enter a character ");
8.  ch=getch(); // taking an user input without printing the value.
9.  printf("\nvalue of ch is %c",ch);
10.       printf("\nEnter a character again ");
11.       ch=getche(); // taking an user input and then displaying it on the screen.
12.        printf("\nvalue of ch is %c",ch);
13.       return 0;
```

```
14.     }
```

Output:

```
Enter a character
value of ch is a
Enter a character again a
value of ch is a
```

In the above example, the value entered through a getch() function is not displayed on the screen while the value entered through a getche() function is displayed on the screen.

---

33) What is the newline escape sequence?

The new line escape sequence is represented by "\n". It inserts a new line on the output screen.

More details.

---

34) Who is the main contributor in designing the C language after Dennis Ritchie?

Brain Kernighan.

---

35) What is the difference between near, far and huge pointers?

A virtual address is composed of the *selector* and *offset*.

A **near** pointer doesn't have explicit selector whereas **far, and huge** pointers have explicit selector. When you perform pointer arithmetic on the far pointer, the selector is not modified, but in case of a huge pointer, it can be modified.

These are the non-standard keywords and implementation specific. These are irrelevant in a modern platform.

---

36) What is the maximum length of an identifier?

It is 32 characters ideally but implementation specific.

---

37) What is typecasting?

The typecasting is a process of converting one data type into another is known as typecasting. If we want to store the floating type value to an int type, then we will convert the data type into another data type explicitly.

Syntax

```
1. (type_name) expression;
```

---

38) What are the functions to open and close the file in C language?

The *fopen()* function is used to open file whereas *fclose()* is used to close file.

---

39) Can we access the array using a pointer in C language?

Yes, by holding the base address of array into a pointer, we can access the array using a pointer.

---

40) What is an infinite loop?

A loop running continuously for an indefinite number of times is called the infinite loop.

Infinite For Loop:

```
1. for(;;){
2. //code to be executed
3. }
```

Infinite While Loop:

```
1. while(1){
2. //code to be executed
3. }
```

Infinite Do-While Loop:

```
1. do{
2. //code to be executed
3. }while(1);
```

---

41) Write a program to print "hello world" without using a semicolon?

```
1. #include<stdio.h>
2. void main(){
3.  if(printf("hello world")){} // It prints the ?hello world? on the screen.
4. }
```

More details.

---

42) Write a program to swap two numbers without using the third variable?

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

```
2. #include<conio.h>
3. main()
4. {
5. int a=10, b=20;    //declaration of variables.
6. clrscr();        //It clears the screen.
7. printf("Before swap a=%d b=%d",a,b);
8.
9. a=a+b;//a=30 (10+20)
10.      b=a-b;//b=10 (30-20)
11.      a=a-b;//a=20 (30-10)
12.
13.      printf("\nAfter swap a=%d b=%d",a,b);
14.      getch();
15.      }
```

More details.

---

43) Write a program to print Fibonacci series without using recursion?

```
1. #include<stdio.h>
2. #include<conio.h>
3. void main()
```

```
{
 int n1=0,n2=1,n3,i,number;
 clrscr();
 printf("Enter the number of elements:");
 scanf("%d",&number);
 printf("\n%d %d",n1,n2);//printing 0 and 1

      for(i=2;i<number;++i)//loop starts from 2 because 0 and 1 are already printed
      {
       n3=n1+n2;
       printf(" %d",n3);
       n1=n2;
       n2=n3;
      }
     getch();
     }
```

More details.

---

44) Write a program to print Fibonacci series using recursion?

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

```
2. #include<conio.h>
3. void printFibonacci(int n) // function to calculate the fibonacci series of a given number.
4. {
5. static int n1=0,n2=1,n3;    // declaration of static variables.
6.     if(n>0){
7.          n3 = n1 + n2;
8.          n1 = n2;
9.         n2 = n3;
10.              printf("%d ",n3);
11.              printFibonacci(n-1);    //calling the function recursively.
12.          }
13.      }
14.      void main(){
15.          int n;
16.          clrscr();
17.          printf("Enter the number of elements: ");
18.          scanf("%d",&n);
19.          printf("Fibonacci Series: ");
20.          printf("%d %d ",0,1);
21.          printFibonacci(n-2);//n-2 because 2 numbers are already printed
22.          getch();
```

```
23.     }
```

More details.

---

45) Write a program to check prime number in C Programming?

```
1. #include<stdio.h>
2. #include<conio.h>
3. void main()
4. {
5. int n,i,m=0,flag=0;    //declaration of variables.
6. clrscr();    //It clears the screen.
7. printf("Enter the number to check prime:");
8. scanf("%d",&n);
9. m=n/2;
10.     for(i=2;i<=m;i++)
11.     {
12.     if(n%i==0)
13.     {
14.     printf("Number is not prime");
15.     flag=1;
16.     break;    //break keyword used to terminate from the loop.
```

```
17.     }
18.     }
19.     if(flag==0)
20.     printf("Number is prime");
21.     getch();    //It reads a character from the keyword.
22.     }
```

More details.

---

46) Write a program to check palindrome number in C Programming?

```
1. #include<stdio.h>
2. #include<conio.h>
3. main()
4. {
5. int n,r,sum=0,temp;
6. clrscr();
7. printf("enter the number=");
8. scanf("%d",&n);
9. temp=n;
10.     while(n>0)
11.     {
```

```
12.     r=n%10;
13.     sum=(sum*10)+r;
14.     n=n/10;
15.     }
16.     if(temp==sum)
17.     printf("palindrome number ");
18.     else
19.     printf("not palindrome");
20.     getch();
21.     }
```

More details.

---

47) Write a program to print factorial of given number without using recursion?

```
1. #include<stdio.h>
2. #include<conio.h>
3. void main(){
4.   int i,fact=1,number;
5.   clrscr();
6.   printf("Enter a number: ");
7.   scanf("%d",&number);
```

```
8.
9.   for(i=1;i<=number;i++){
10.            fact=fact*i;
11.        }
12.        printf("Factorial of %d is: %d",number,fact);
13.        getch();
14.      }
```

More details.

---

48) Write a program to print factorial of given number using recursion?

```
1. #include<stdio.h>
2. #include<conio.h>
3.  long factorial(int n)    // function to calculate the factorial of a given number.
4. {
5.   if (n == 0)
6.     return 1;
7. else
8. return(n * factorial(n-1));    //calling the function recursively.
9. }
10.       void main()
```

```
11.     {
12.       int number;    //declaration of variables.
13.       long fact;
14.      clrscr();
15.       printf("Enter a number: ");
16.     scanf("%d", &number);
17.      fact = factorial(number);    //calling a function.
18.     printf("Factorial of %d is %ld\n", number, fact);
19.      getch();   //It reads a character from the keyword.
20.     }
```

More details.

---

49) Write a program to check Armstrong number in C?

```
1. #include<stdio.h>
2. #include<conio.h>
3. main()
4. {
5. int n,r,sum=0,temp;    //declaration of variables.
6. clrscr(); //It clears the screen.
7. printf("enter the number=");
```

```
8. scanf("%d",&n);
9. temp=n;
10.     while(n>0)
11.     {
12.     r=n%10;
13.     sum=sum+(r*r*r);
14.     n=n/10;
15.     }
16.     if(temp==sum)
17.     printf("armstrong  number ");
18.     else
19.     printf("not armstrong number");
20.     getch();  //It reads a character from the keyword.
21.     }
```

More details.

---

50) Write a program to reverse a given number in C?

```
1. #include<stdio.h>
2. #include<conio.h>
3. main()
```

```
4. {
5. int n, reverse=0, rem;    //declaration of variables.
6. clrscr(); // It clears the screen.
7. printf("Enter a number: ");
8. scanf("%d", &n);
9. while(n!=0)
10.     {
11.         rem=n%10;
12.         reverse=reverse*10+rem;
13.         n/=10;
14.     }
15.     printf("Reversed Number: %d",reverse);
16.     getch();  // It reads a character from the keyword.
17.     }
```

Q #1) What are the key features in the C programming language?

Answer: Features are as follows:

- **Portability**: It is a platform-independent language.
- **Modularity:** Possibility to break down large programs into small modules.
- **Flexibility:** The possibility of a programmer to control the language.

- **Speed:** C comes with support for system programming and hence it compiles and executes with high speed when compared with other high-level languages.
- **Extensibility**: Possibility to add new features by the programmer.

Q #2) What are the basic data types associated with C?

Answer:

- **Int –** Represent the number (integer)
- **Float –** Number with a fraction part.
- **Double –** Double-precision floating-point value
- **Char –** Single character
- **Void –** Special purpose type without any value.

Q #3) What is the description for syntax errors?

Answer: The mistakes/errors that occur while creating a program are called syntax errors. Misspelled commands or incorrect case commands, an incorrect number of parameters in calling method /function, data type mismatches can be identified as common examples for syntax errors.

Q #4) What is the process to create increment and decrement statement in C?

Answer: There are two possible methods to perform this task.

- Use increment (++) and decrement (-) operator.

Example When x=4, x++ returns 5 and x- returns 3.

- Use conventional + or – sign.

Example When x=4, use x+1 to get 5 and x-1 to get 3.

Q #5) What are reserved words with a programming language?

Answer: The words that are a part of the standard C language library are called **reserved words**. Those reserved words have special meaning and it is not possible to use them for any activity other than its intended functionality.

Example: void, return int.

Q #6) What is the explanation for the dangling pointer in C?

Answer: When there is a pointer pointing to a memory address of any variable, but after some time the variable was deleted from the memory location while keeping the pointer pointing to that location is known as a dangling pointer in C.

Q #7) Describe static function with its usage?

Answer: A function, which has a function definition prefixed with a static keyword is defined as a static function. The static function should be called within the same source code.

Q #8) What is the difference between abs() and fabs() functions?

Answer: Both functions are to retrieve absolute value. abs() is for integer values and fabs() is for floating type numbers. Prototype for abs() is under the library file < stdlib.h > and fabs() is under < math.h >.

Q #9) Describe Wild Pointers in C?

Answer: Uninitialized pointers in the C code are known as **Wild Pointers**. They point to some arbitrary memory location and can cause bad program behavior or program crash.

Q #10) What is the difference between ++a and a++?

Answer: ‘++a” is called prefixed increment and the increment will happen first on a variable. ‘a++' is called postfix increment and the increment happens after the value of a variable used for the operations.

Q #11) Describe the difference between = and == symbols in C programming?

Answer: ‘==' is the comparison operator which is used to compare the value or expression on the left-hand side with the value or expression on the right-hand side.

‘=' is the assignment operator which is used to assign the value of the right-hand side to the variable on the left-hand side.

Q #12) What is the explanation for prototype function in C?

Answer: Prototype function is a declaration of a function with the following information to the compiler.

- Name of the function.
- The return type of the function.
- Parameters list of the function.

```
int Sum(int, int);
```

In this example Name of the function is Sum, the return type is the integer data type and it accepts two integer parameters.

Q #13) What is the explanation for the cyclic nature of data types in C?

Answer: Some of the data types in C have special characteristic nature when a developer assigns value beyond the range of the data type. There will be no compiler error and the value changes

according to a cyclic order. This is called cyclic nature. Char, int, long int data types have this property. Further float, double and long double data types do not have this property.

Q #14) Describe the header file and its usage in C programming?

Answer: The file containing the definitions and prototypes of the functions being used in the program are called a header file. It is also known as a library file.

Example: The header file contains commands like printf and scanf is from the stdio.h library file.

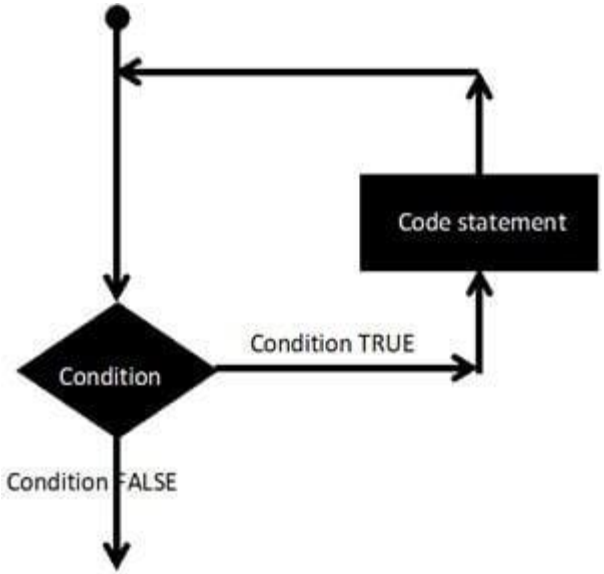
Q #15) There is a practice in coding to keep some code blocks in comment symbols than delete it when debugging. How this affects when debugging?

Answer: This concept is called commenting out and this is the way to isolate some part of the code which scans possible reason for the error. Also, this concept helps to save time because if the code is not the reason for the issue it can simply be removed from comment.

Q #16) What are the general description for loop statements and available loop types in C?

Answer: A statement that allows the execution of statements or groups of statements in a repeated way is defined as a loop.

The following diagram explains a general form of a loop.



There are 4 types of loop statements in C.

- **While loop**
- **For Loop**
- **Do…While Loop**
- **Nested Loop**

Q #17) What is a nested loop?

Answer: A loop that runs within another loop is referred to as a **nested loop**. The first loop is called the Outer Loop and the inside loop is called the Inner Loop. The inner loop executes the number of times defined in an outer loop.

Q #18) What is the general form of function in C?

Answer: The function definition in C contains four main sections.

return_type function_name(parameter list)

```
{
    body of the function
}
```

- **Return Type**: Data type of the return value of the function.
- **Function Name:** The name of the function and it is important to have a meaningful name that describes the activity of the function.
- **Parameters**: The input values for the function that are used to perform the required action.
- **Function Body**: Collection of statements that performs the required action.

Q #19) What is a pointer on a pointer in C programming language?

Answer: A pointer variable that contains the address of another pointer variable is called pointer on a pointer. This concept de-refers twice to point to the data held by a pointer variable.

```
int a = 5, *x=&a, **y=&x;
```

In this example **y returns the value of the variable a.

Q #20) What are the valid places to have keyword "Break"?

Answer: The purpose of the Break keyword is to bring the control out of the code block which is executing. It can appear only in looping or switch statements.

Q #21) What is the behavioral difference when the header file is included in double-quotes ("") and angular braces (<>)?

Answer: When the Header file is included within double quotes (" "), compiler search first in the working directory for the particular header file. If not found, then it searches the file in the include

path. But when the Header file is included within angular braces (<>), the compiler only searches in the working directory for the particular header file.

Q #22) What is a sequential access file?

Answer: General programs store data into files and retrieve existing data from files. With the sequential access file, such data are saved in a sequential pattern. When retrieving data from such files each data is read one by one until the required information is found.

Q #23) What is the method to save data in a stack data structure type?

Answer: Data is stored in the Stack data structure type using the **First In Last Out (FILO)** mechanism. Only top of the stack is accessible at a given instance. Storing mechanism is referred as a PUSH and retrieve is referred to as a POP.

Q #24) What is the significance of C program algorithms?

Answer: The algorithm is created first and it contains step by step guidelines on how the solution should be. Also, it contains the steps to consider and the required calculations/operations within the program.

Q #25) What is the correct code to have the following output in C using nested for loop?

Answer:

```
#include <stdio.h>

int main () {

    int a;
    int b;
    /* for loop execution */
    for( a = 1; a < 6; a++ )
    {
        /* for loop execution */
        for ( b = 1; b <= a; b++ )
        {
            printf("%d",b);
         }
         printf("\n");
    }

    return 0;
}
```

Q #26) Explain the use of function toupper() with an example code?

Answer: Toupper() function is used to convert the value to uppercase when it used with characters.

Code:

```
#include <stdio.h>
#include <ctype.h>
int main()
{
    char c;

    c = 'a';
    printf("%c -> %c", c, toupper(c));
```

```
    c = 'A';
    printf("\n%c -> %c", c, toupper(c));

    c = '9';
    printf("\n%c -> %c", c, toupper(c));
    return 0;
}
```

Result:

Q #27) What is the code in a while loop that returns the output of the given code?

```
#include <stdio.h>

int main () {

    int a;

    /* for loop execution */
    for( a = 1; a <= 100; a++ )
    {
```

```
        printf("%d\n",a * a);
     }

     return 0;
}
```

Answer:

```
#include <stdio.h>

int main () {

    int a;

    while (a<=100)
    {
```

```
        printf ("%d\n", a * a);
        a++;
    }
    return 0;
}
```

Q #28) Select the incorrect operator form in the following list(== , <> , >= , <=) and what is the reason for the answer?

Answer: Incorrect operator is '<>'. This format is correct when writing conditional statements, but it is not the correct operation to indicate not equal in C programming. It gives a compilation error as follows.

Code:

```
#include <stdio.h>

int main () {
```

```
    if ( 5 <> 10 )
        printf( "test for <>" );
      return 0;
}
```

Error:

Q #29) Is it possible to use curly brackets ({}) to enclose a single line code in C program?

Answer: Yes, it works without any error. Some programmers like to use this to organize the code. But the main purpose of curly brackets is to group several lines of codes.

Q #30) Describe the modifier in C?

Answer: Modifier is a prefix to the basic data type which is used to indicate the modification for storage space allocation to a variable.

Example– In a 32-bit processor, storage space for the int data type is 4.When we use it with modifier the storage space change as follows:

- **Long int:** Storage space is 8 bit
- **Short int:** Storage space is 2 bit

Q #31) What are the modifiers available in C programming language?

Answer: There are 5 modifiers available in the C programming language as follows:

- Short
- Long
- Signed
- Unsigned
- long long

Q #32) What is the process to generate random numbers in C programming language?

Answer: The command rand() is available to use for this purpose. The function returns an integer number beginning from zero(0). The following sample code demonstrates the use of rand().

Code:

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

int main ()
{
int a;
```

```
int b;


    for(a=1; a<11; a++)
   {
        b = rand();
        printf( "%d\n", b );
    }
      return 0;
}
```

Output:

Q #33) Describe the newline escape sequence with a sample program?

Answer: The Newline escape sequence is represented by \n. This indicates the point that the new line starts to the compiler and the output is created accordingly. The following sample program demonstrates the use of the newline escape sequence.

Code:

```
/*
* C Program to print string
*/
#include <stdio.h>
#include <string.h>

int main(){
    printf("String 01 ");
    printf("String 02 ");
```

```
    printf("String 03 \n");

    printf("String 01 \n");
    printf("String 02 \n");
    return 0;
}
```

Output:

Q #34) Is that possible to store 32768 in an int data type variable?

Answer: Int data type is only capable of storing values between – 32768 to 32767. To store 32768 a modifier needs to used with the int data type. Long Int can use and also if there are no negative values, unsigned int is also possible to use.

Q #35) Is there any possibility to create a customized header file with C programming language?

Answer: Yes, it is possible and easy to create a new header file. Create a file with function prototypes that are used inside the program. Include the file in the `#include' section from its name.

Q #36) Describe dynamic data structure in C programming language?

Answer: Dynamic data structure is more efficient to memory. The memory access occurs as needed by the program.

Q #37) Is that possible to add pointers to each other?

Answer: There is no possibility to add pointers together. Since pointer contains address details there is no way to retrieve the value from this operation.

Q #38) What is indirection?

Answer: If you have defined a pointer to a variable or any memory object, there is no direct reference to the value of the variable. This is called the indirect reference. But when we declare a variable, it has a direct reference to the value.

Q #39) What are the ways to a null pointer that can be used in the C programming language?

Answer: Null pointers are possible to use in three ways.

- As an error value.
- As a sentinel value.
- To terminate indirection in the recursive data structure.

Q #40) What is the explanation for modular programming?

Answer: The process of dividing the main program into executable subsection is called module programming. This concept promotes reusability.

C++ Interview Questions

A list of top frequently asked C++ interview questions and answers are given below.

1) What is C++?

C++ is an object-oriented programming language created by Bjarne Stroustrup. It was released in 1985.

C++ is a superset of C with the major addition of classes in C language.

Initially, Stroustrup called the new language "C with classes". However, after sometime the name was changed to C++. The idea of C++ comes from the C increment operator ++.

---

2) What are the advantages of C++?

C++ doesn't only maintains all aspects from C language, it also simplifies memory management and adds several features like:

- C++ is a highly portable language means that the software developed using C++ language can run on any platform.
- C++ is an object-oriented programming language which includes the concepts such as classes, objects, inheritance, polymorphism, abstraction.

- C++ has the concept of inheritance. Through inheritance, one can eliminate the redundant code and can reuse the existing classes.
- Data hiding helps the programmer to build secure programs so that the program cannot be attacked by the invaders.
- Message passing is a technique used for communication between the objects.
- C++ contains a rich function library.

---

3) What is the difference between C and C++?

Following are the differences between C and C++:

C	C++
C language was developed by Dennis Ritchie.	C++ language was developed by Bjarne Stroustrup.
C is a structured programming language.	C++ supports both structural and object-oriented programming language.
C is a subset of C++.	C++ is a superset of C.

In C language, data and functions are the free entities.	In the C++ language, both data and functions are encapsulated together in the form of a project.
C does not support the data hiding. Therefore, the data can be used by the outside world.	C++ supports data hiding. Therefore, the data cannot be accessed by the outside world.
C supports neither function nor operator overloading.	C++ supports both function and operator overloading.
In C, the function cannot be implemented inside the structures.	In the C++, the function can be implemented inside the structures.
Reference variables are not supported in C language.	C++ supports the reference variables.
C language does not support the virtual and friend functions.	C++ supports both virtual and friend functions.
In C, scanf() and printf() are mainly used for input/output.	C++ mainly uses stream cin and cout to perform input and output operations.

4) What is the difference between reference and pointer?

Following are the differences between reference and pointer:

Reference	Pointer
Reference behaves like an alias for an existing variable, i.e., it is a temporary variable.	The pointer is a variable which stores the address of a variable.
Reference variable does not require any indirection operator to access the value. A reference variable can be used directly to access the value.	Pointer variable requires an indirection operator to access the value of a variable.
Once the reference variable is assigned, then it cannot be reassigned with different address values.	The pointer variable is an independent variable means that it can be reassigned to point to different objects.
A null value cannot be assigned to the reference variable.	A null value can be assigned to the reference variable.

It is necessary to initialize the variable at the time of declaration.	It is not necessary to initialize the variable at the time of declaration.

---

5) What is a class?

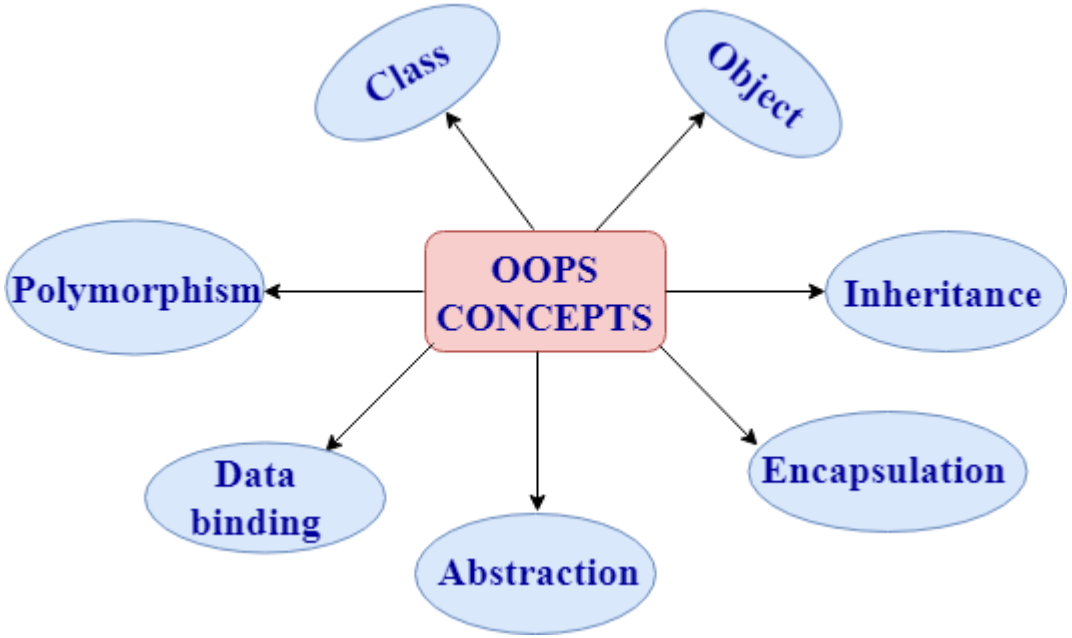
The class is a user-defined data type. The class is declared with the keyword class. The class contains the data members, and member functions whose access is defined by the three modifiers are private, public and protected. The class defines the type definition of the category of things. It defines a datatype, but it does not define the data it just specifies the structure of data.

You can create N number of objects from a class.

---

6) What are the various OOPs concepts in C++?

The various OOPS concepts in C++ are:



- **Class:**

The class is a user-defined data type which defines its properties and its functions. For example, Human being is a class. The body parts of a human being are its properties, and the actions performed by the body parts are known as functions. The class does not occupy any memory space. Therefore, we can say that the class is the only logical representation of the data.

The syntax of declaring the class:

```
1. class student
2. {
3. //data members;
```

```
4. //Member functions
5. }
```

- **Object:**

An object is a run-time entity. An object is the instance of the class. An object can represent a person, place or any other item. An object can operate on both data members and member functions. The class does not occupy any memory space. When an object is created using a new keyword, then space is allocated for the variable in a heap, and the starting address is stored in the stack memory. When an object is created without a new keyword, then space is not allocated in the heap memory, and the object contains the null value in the stack.

```
1. class Student
2. {
3. //data members;
4. //Member functions
5. }
```

The syntax for declaring the object:

```
1. Student s = new Student();
```

- **Inheritance:**

Inheritance provides reusability. Reusability means that one can use the functionalities of the existing class. It eliminates the redundancy of code. Inheritance is a technique of deriving a new class from the old class. The old class is known as the base class, and the new class is known as derived class.

Syntax

1. **class** derived_class :: visibility-mode base_class;

Note: The visibility-mode can be public, private, protected.

- **Encapsulation:**

Encapsulation is a technique of wrapping the data members and member functions in a single unit. It binds the data within a class, and no outside method can access the data. If the data member is private, then the member function can only access the data.

- **Abstraction:**

Abstraction is a technique of showing only essential details without representing the implementation details. If the members are defined with a public keyword, then the members are accessible outside also. If the members are defined with a private keyword, then the members are not accessible by the outside methods.

- **Data binding:**

Data binding is a process of binding the application UI and business logic. Any change made in the business logic will reflect directly to the application UI.

- **Polymorphism:**

Polymorphism means multiple forms. Polymorphism means having more than one function with the same name but with different functionalities. Polymorphism is of two types:

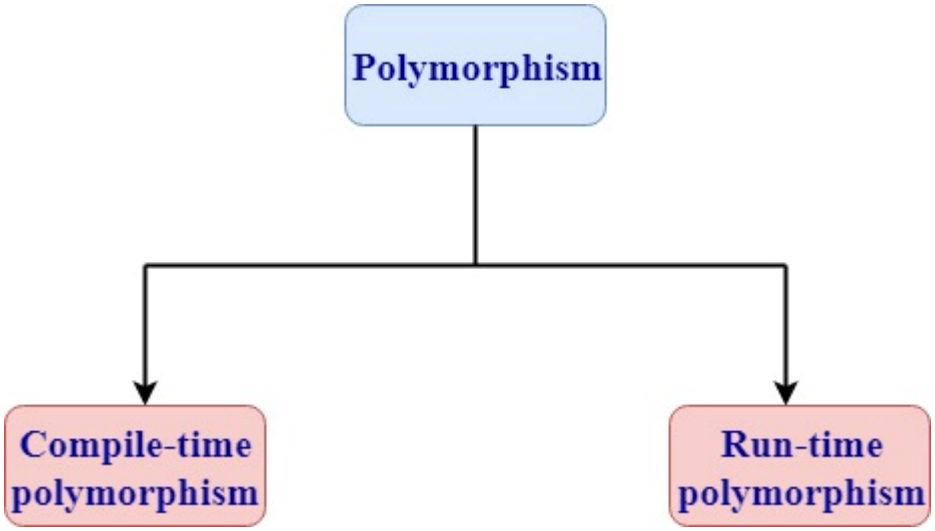
1. Static polymorphism is also known as early binding.
2. Dynamic polymorphism is also known as late binding.

---

7) What are the different types of polymorphism in C++?

Polymorphism: Polymorphism means multiple forms. It means having more than one function with the same function name but with different functionalities.

Polymorphism is of two types:



- **Runtime polymorphism**

Runtime polymorphism is also known as dynamic polymorphism. Function overriding is an example of runtime polymorphism. Function overriding means when the child class contains the method which is already present in the parent class. Hence, the child class overrides the method of the parent class. In case of function overriding, parent and child class both contains the same function with the different definition. The call to the function is determined at runtime is known as runtime polymorphism.

Let's understand this through an example:

```
1. #include <iostream>
2. using namespace std;
3. class Base
```

```
4. {
5.     public:
6.     virtual void show()
7.     {
8.         cout<<"javaTpoint";
9.      }
10.       };
11.       class Derived:public Base
12.       {
13.          public:
14.          void show()
15.          {
16.              cout<<"javaTpoint tutorial";
17.          }
18.       };
19.
20.       int main()
21.       {
22.          Base* b;
23.          Derived d;
24.          b=&d;
```

```
25.         b->show();
26.                 return 0;
27.     }
```

Output:

```
javaTpoint tutorial
```

- **Compile time polymorphism**

Compile-time polymorphism is also known as static polymorphism. The polymorphism which is implemented at the compile time is known as compile-time polymorphism. Method overloading is an example of compile-time polymorphism.

Method overloading: Method overloading is a technique which allows you to have more than one function with the same function name but with different functionality.

Method overloading can be possible on the following basis:

- The return type of the overloaded function.
- The type of the parameters passed to the function.
- The number of parameters passed to the function.

Let's understand this through an example:

```
1. #include <iostream>
2. using namespace std;
3. class Multiply
4. {
5.    public:
6.    int mul(int a,int b)
7.    {
8.       return(a*b);
9.    }
10.        int mul(int a,int b,int c)
11.        {
12.           return(a*b*c);
13.       }
14.      };
15.      int main()
16.      {
17.         Multiply multi;
18.         int res1,res2;
19.         res1=multi.mul(2,3);
20.         res2=multi.mul(2,3,4);
21.         cout<<"\n";
```

```
22.         cout<<res1;
23.         cout<<"\n";
24.         cout<<res2;
25.         return 0;
26.     }
```

Output:

```
6
24
```

- In the above example, mul() is an overloaded function with the different number of parameters.

---

8) Define namespace in C++.

- The namespace is a logical division of the code which is designed to stop the naming conflict.
- The namespace defines the scope where the identifiers such as variables, class, functions are declared.
- The main purpose of using namespace in C++ is to remove the ambiguity. Ambiquity occurs when the different task occurs with the same name.
- For example: if there are two functions exist with the same name such as add(). In order to prevent this ambiguity, the namespace is used. Functions are declared in different namespaces.

- C++ consists of a standard namespace, i.e., std which contains inbuilt classes and functions. So, by using the statement "using namespace std;" includes the namespace "std" in our program.
- **Syntax of namespace:**

```
1. namespace namespace_name
2. {
3.  //body of namespace;
4. }
```

Syntax of accessing the namespace variable:

```
1. namespace_name::member_name;
```

Let's understand this through an example:

```
1. #include <iostream>
2. using namespace std;
3. namespace addition
4. {
5.     int a=5;
6.     int b=5;
7.     int add()
```

```
8.    {
9.        return(a+b);
10.         }
11.      }
12.
13.      int main() {
14.      int result;
15.      result=addition::add();
16.      cout<<result;
17.      return 0;
18.      }
```

Output:

```
10
```

---

9) Define token in C++.

A token in C++ can be a keyword, identifier, literal, constant and symbol.

---

10) Who was the creator of C++?

Bjarne Stroustrup.

---

11) Which operations are permitted on pointers?

Following are the operations that can be performed on pointers:

- **Incrementing or decrementing a pointer**: Incrementing a pointer means that we can increment the pointer by the size of a data type to which it points.

There are two types of increment pointers:

1. Pre-increment pointer: The pre-increment operator increments the operand by 1, and the value of the expression becomes the resulting value of the incremented. Suppose ptr is a pointer then pre-increment pointer is represented as ++ptr.

Let's understand this through an example:

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int a[5]={1,2,3,4,5};
6. int *ptr;
```

```
7. ptr=&a[0];
8. cout<<"Value of *ptr is : "<<*ptr<<"\n";
9. cout<<"Value of *++ptr : "<<*++ptr;
10.     return 0;
11.     }
```

Output:

```
Value of *ptr is : 1
Value of *++ptr : 2
```

2. Post-increment pointer: The post-increment operator increments the operand by 1, but the value of the expression will be the value of the operand prior to the incremented value of the operand. Suppose ptr is a pointer then post-increment pointer is represented as ptr++.

Let's understand this through an example:

```
1. #include <iostream>
2. using namespace std;
3. int main()
4. {
5. int a[5]={1,2,3,4,5};
6. int *ptr;
```

```
7. ptr=&a[0];
8. cout<<"Value of *ptr is : "<<*ptr<<"\n";
9. cout<<"Value of *ptr++ : "<<*ptr++;
10.     return 0;
11.     }
```

Output:

```
Value of *ptr is : 1
Value of *ptr++ : 1
```

- **Subtracting a pointer from another pointer:** When two pointers pointing to the members of an array are subtracted, then the number of elements present between the two members are returned.

---

12) Define 'std'.

Std is the default namespace standard used in C++.

---

13) Which programming language's unsatisfactory performance led to the discovery of C++?

C++was discovered in order to cope with the disadvantages of C.

---

14) How delete [] is different from delete?

Delete is used to release a unit of memory, delete[] is used to release an array.

---

15) What is the full form of STL in C++?

STL stands for Standard Template Library.

---

16) What is an object?

The Object is the instance of a class. A class provides a blueprint for objects. So you can create an object from a class. The objects of a class are declared with the same sort of declaration that we declare variables of basic types.

---

17) What are the C++ access specifiers?

The access specifiers are used to define how to functions and variables can be accessed outside the class.

There are three types of access specifiers:

- **Private**: Functions and variables declared as private can be accessed only within the same class, and they cannot be accessed outside the class they are declared.
- **Public**: Functions and variables declared under public can be accessed from anywhere.
- **Protected**: Functions and variables declared as protected cannot be accessed outside the class except a child class. This specifier is generally used in inheritance.

---

18) What is Object Oriented Programming (OOP)?

OOP is a methodology or paradigm that provides many concepts. The basic concepts of Object Oriented Programming are given below:

Classes and Objects: Classes are used to specify the structure of the data. They define the data type. You can create any number of objects from a class. Objects are the instances of classes.

Encapsulation: Encapsulation is a mechanism which binds the data and associated operations together and thus hides the data from the outside world. Encapsulation is also known as data hiding. In C++, It is achieved using the access specifiers, i.e., public, private and protected.

Abstraction: Abstraction is used to hide the internal implementations and show only the necessary details to the outer world. Data abstraction is implemented using interfaces and abstract classes in C++.

Some people confused about Encapsulation and abstraction, but they both are different.

Inheritance: Inheritance is used to inherit the property of one class into another class. It facilitates you to define one class in term of another class.

---

19) What is the difference between an array and a list?

- An Array is a collection of homogeneous elements while a list is a collection of heterogeneous elements.
- Array memory allocation is static and continuous while List memory allocation is dynamic and random.
- In Array, users don't need to keep in track of next memory allocation while In the list, the user has to keep in track of next location where memory is allocated.

---

20) What is the difference between new() and malloc()?

- new() is a preprocessor while malloc() is a function.
- There is no need to allocate the memory while using "new" but in malloc() you have to use sizeof().
- "new" initializes the new memory to 0 while malloc() gives random value in the newly allotted memory location.
- The new() operator allocates the memory and calls the constructor for the object initialization and malloc() function allocates the memory but does not call the constructor for the object initialization.
- The new() operator is faster than the malloc() function as operator is faster than the function.

---

21) What are the methods of exporting a function from a DLL?

There are two ways:

- By using the DLL's type library.
- Taking a reference to the function from the DLL instance.

---

22) Define friend function.

Friend function acts as a friend of the class. It can access the private and protected members of the class. The friend function is not a member of the class, but it must be listed in the class definition. The non-member function cannot access the private data of the class. Sometimes, it is necessary for the non-member function to access the data. The friend function is a non-member function and has the ability to access the private data of the class.

To make an outside function friendly to the class, we need to declare the function as a friend of the class as shown below:

```
1. class sample
2. {
3.    // data members;
4.  public:
5. friend void abc(void);
6. };
```

Following are the characteristics of a friend function:

- The friend function is not in the scope of the class in which it has been declared.
- Since it is not in the scope of the class, so it cannot be called by using the object of the class. Therefore, friend function can be invoked like a normal function.
- A friend function cannot access the private members directly, it has to use an object name and dot operator with each member name.

- Friend function uses objects as arguments.

Let's understand this through an example:

```
1. #include <iostream>
2. using namespace std;
3. class Addition
4. {
5.  int a=5;
6.  int b=6;
7.  public:
8.  friend int add(Addition a1)
9.  {
10.          return(a1.a+a1.b);
11.       }
12.      };
13.      int main()
14.      {
15.      int result;
16.      Addition a1;
17.       result=add(a1);
18.       cout<<result;
```

```
19.     return 0;
20.     }
```

Output:

```
11
```

---

23) What is a virtual function?

- A virtual function is used to replace the implementation provided by the base class. The replacement is always called whenever the object in question is actually of the derived class, even if the object is accessed by a base pointer rather than a derived pointer.
- A virtual function is a member function which is present in the base class and redefined by the derived class.
- When we use the same function name in both base and derived class, the function in base class is declared with a keyword virtual.
- When the function is made virtual, then C++ determines at run-time which function is to be called based on the type of the object pointed by the base class pointer. Thus, by making the base class pointer to point different objects, we can execute different versions of the virtual functions.

Rules of a virtual function:

- The virtual functions should be a member of some class.
- The virtual function cannot be a static member.
- Virtual functions are called by using the object pointer.
- It can be a friend of another class.
- C++ does not contain virtual constructors but can have a virtual destructor.

---

24) When should we use multiple inheritance?

You can answer this question in three manners:

1. Never
2. Rarely
3. If you find that the problem domain cannot be accurately modeled any other way.

---

25) What is a destructor?

A Destructor is used to delete any extra resources allocated by the object. A destructor function is called automatically once the object goes out of the scope.

Rules of destructor:

- Destructors have the same name as class name and it is preceded by tilde.
- It does not contain any argument and no return type.

---

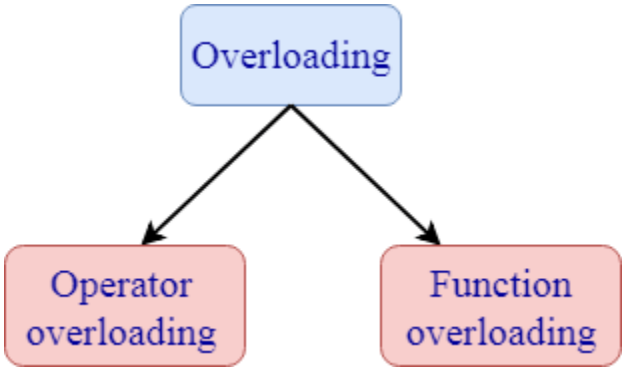
26) What is an overflow error?

It is a type of arithmetical error. It happens when the result of an arithmetical operation been greater than the actual space provided by the system.

---

27) What is overloading?

- When a single object behaves in many ways is known as overloading. A single object has the same name, but it provides different versions of the same function.
- C++ facilitates you to specify more than one definition for a function name or an operator in the same scope. It is called function overloading and operator overloading respectively.
- **Overloading is of two types:**



1. Operator overloading: Operator overloading is a compile-time polymorphism in which a standard operator is overloaded to provide a user-defined definition to it. For example, '+' operator is overloaded to perform the addition operation on data types such as int, float, etc.

Operator overloading can be implemented in the following functions:

- Member function
- Non-member function
- Friend function

Syntax of Operator overloading:

```
1. Return_type classname :: Operator Operator_symbol(argument_list)
2. {
3.       // body_statements;
4. }
```

2. Function overloading: Function overloading is also a type of compile-time polymorphism which can define a family of functions with the same name. The function would perform different operations based on the argument list in the function call. The function to be invoked depends on the number of arguments and the type of the arguments in the argument list.

---

28) What is function overriding?

If you inherit a class into a derived class and provide a definition for one of the base class's function again inside the derived class, then this function is called overridden function, and this mechanism is known as function overriding.

---

29) What is virtual inheritance?

Virtual inheritance facilitates you to create only one copy of each object even if the object appears more than one in the hierarchy.

---

30) What is a constructor?

A Constructor is a special method that initializes an object. Its name must be same as class name.

31) What is the purpose of the "delete" operator?

The "delete" operator is used to release the dynamic memory created by "new" operator.

---

32) Explain this pointer?

This pointer holds the address of the current object.

---

33) What does Scope Resolution operator do?

A scope resolution operator(::) is used to define the member function outside the class.

---

34) What is the difference between delete and delete[]?

Delete [] is used to release the array of allocated memory which was allocated using new[] whereas delete is used to release one chunk of memory which was allocated using new.

---

35) What is a pure virtual function?

The pure virtual function is a virtual function which does not contain any definition. The normal function is preceded with a keyword virtual. The pure virtual function ends with 0.

Syntax of a pure virtual function:

```
1. virtual void abc()=0;   //pure virtual function.
```

Let's understand this through an example:

```
1. #include<iostream>
2. using namespace std;
3. class Base
4. {
5.     public:
6.     virtual void show()=0;
7. };
8.
9. class Derived:public Base
10.     {
11.         public:
12.         void show()
```

```
13.         {
14.             cout<<"javaTpoint";
15.         }
16.     };
17.     int main()
18.     {
19.         Base* b;
20.         Derived d;
21.         b=&d;
22.         b->show();
23.         return 0;
24.     }
```

Output:

```
javaTpoint
```

---

36) What is the difference between struct and class?

Structures	class

A structure is a user-defined data type which contains variables of dissimilar data types.	The class is a user-defined data type which contains member variables and member functions.
The variables of a structure are stored in the stack memory.	The variables of a class are stored in the heap memory.
We cannot initialize the variables directly.	We can initialize the member variables directly.
If access specifier is not specified, then by default the access specifier of the variable is "public".	If access specifier is not specified, then by default the access specifier of a variable is "private".
The instance of a structure is a "structure variable".	

Declaration of a structure:
struct structure_name
{
// body of structure;
	Declaration of class:
	class class_name
	{
	// body of class;

} ;	}
A structure is declared by using a struct keyword.	The class is declared by using a class keyword.
The structure does not support the inheritance.	The class supports the concept of inheritance.
The type of a structure is a value type.	The type of a class is a reference type.

---

37) What is a class template?

A class template is used to create a family of classes and functions. For example, we can create a template of an array class which will enable us to create an array of various types such as int, float, char, etc. Similarly, we can create a template for a function, suppose we have a function add(), then we can create multiple versions of add().

The syntax of a class template:

```
1. template<class T>
2. class classname
3. {
4.   // body of class;
```

5. };

Syntax of a object of a template class:

1. classname<type> objectname(arglist);

---

38) What is the difference between function overloading and operator overloading?

Function overloading: Function overloading is defined as we can have more than one version of the same function. The versions of a function will have different signature means that they have a different set of parameters.

Operator overloading: Operator overloading is defined as the standard operator can be redefined so that it has a different meaning when applied to the instances of a class.

---

39) What is a virtual destructor?

A virtual destructor in C++ is used in the base class so that the derived class object can also be destroyed. A virtual destructor is declared by using the ~ tilde operator and then virtual keyword before the constructor.

Note: Constructor cannot be virtual, but destructor can be virtual.

Let's understand this through an example

- Example without using virtual destructor

```
1. #include <iostream>
2. using namespace std;
3. class Base
4. {
5.     public:
6.     Base()
7.     {
8.         cout<<"Base constructor is called"<<"\n";
9.     }
10.         ~Base()
11.         {
12.             cout<<"Base class object is destroyed"<<"\n";
13.         }
14.     };
15.     class Derived:public Base
16.     {
```

```
17.         public:
18.         Derived()
19.         {
20.             cout<<"Derived class constructor is called"<<"\n";
21.         }
22.         ~Derived()
23.         {
24.             cout<<"Derived class object is destroyed"<<"\n";
25.         }
26.     };
27.     int main()
28.     {
29.       Base* b= new Derived;
30.       delete b;
31.       return 0;
32.
33.     }
```

Output:

```
Base constructor is called
Derived class constructor is called
```

```
Base class object is destroyed
```

In the above example, delete b will only call the base class destructor due to which derived class destructor remains undestroyed. This leads to the memory leak.

- Example with a virtual destructor

```
1. #include <iostream>
2. using namespace std;
3. class Base
4. {
5.     public:
6.     Base()
7.     {
8.         cout<<"Base constructor is called"<<"\n";
9.     }
10.         virtual ~Base()
11.         {
12.             cout<<"Base class object is destroyed"<<"\n";
13.         }
14.     };
15.     class Derived:public Base
```

```
16.     {
17.         public:
18.         Derived()
19.         {
20.             cout<<"Derived class constructor is called"<<"\n";
21.         }
22.         ~Derived()
23.         {
24.             cout<<"Derived class object is destroyed"<<"\n";
25.         }
26.     };
27.     int main()
28.     {
29.       Base* b= new Derived;
30.       delete b;
31.       return 0;
32.
33.     }
```

Output:

```
Base constructor is called
```

```
Derived class constructor is called
Derived class object is destroyed
Base class object is destroyed
```

When we use the virtual destructor, then the derived class destructor is called first, and then the base class destructor is called.

Questions and Answers

Object Oriented Programming& Methodology

Q1. What is oops?

Ans: OOPS is abbreviated as Object Oriented Programming system in which programs are considered as a collection of objects. Each object is nothing but an instance of a class.

Q2.Write Basic Concepts Of Oops?

Ans: Following are the concepts of OOPS and are as follows:

1-Abstraction.

2-Encapsulation.

3-Inheritance.

4-Polymorphism.

Q3.what is a class?

Ans: A class is simply a representation of a type of object. It is the blueprint/ plan/ template that describe the details of an object.

Q4.what is an object?

Ans: Object is termed as an instance of a class, and it has its own state, behavior and identity.

Q5.what is encapsulation?

Ans: Encapsulation is an attribute of an object, and it contains all data which is hidden. That hidden data can be restricted to the members of that class.Levels are Public, Protected, Private, Internal and Protected Internal.

Q6.what is polymorphism?

Ans: Polymorphism is nothing but assigning behavior or value in a subclass to something that was already declared in the main class. Simply, polymorphism takes more than one form.

Q7.what is inheritance?

Ans: Inheritance is a concept where one class shares the structure and behavior defined in another class. If inheritance applied on one class is called Single Inheritance, and if it depends on multiple classes, then it is called multiple inheritances.

Q 8.What are manipulators?

Ans: Manipulators are the functions which can be used in conjunction with the insertion (<<) and extraction (>>) operators on an object. Examples are endl and setw.

Q9. Define a constructor?

Ans: Constructor is a method used to initialize the state of an object, and it gets invoked at the time of object creation. Rules for constructor are:.

1-Constructor Name should be same as class name.

2-Constructor must have no return type.

Q10. Define destructor?

Ans: Destructor is a method which is automatically called when the object is made of scope or destroyed. Destructor name is also same as class name but with the tilde symbol before the name.

Q11. What is inline function?

Ans: Inline function is a technique used by the compilers and instructs to insert complete body of the function wherever that function is used in the program source code.

Q12. What is a virtual function?

Ans: Virtual function is a member function of class and its functionality can be overridden in its derived class. This function can be implemented by using a keyword called virtual, and it can be given during function declaration. Virtual function can be achieved in C++, and it can be achieved in C Language by using function pointers or pointers to function.

Q13. What is friend function?

Ans: Friend function is a friend of a class that is allowed to access to Public, private or protected data in that same class. If the function is defined outside the class cannot access such information. Friend can be declared anywhere in the class declaration, and it cannot be affected by access control keywords like private, public or protected.

Q14. What is function overloading?

Ans: Function overloading is defined as a normal function, but it has the ability to perform different tasks. It allows creation of several methods with the same name which differ from each other by type of input and output of the function.

Example:

1-Void add (int& a, int& b);

2-Void add (double& a, double& b);

3-Void add (struct bob& a, struct bob& b);

Q15. What Is Operator Overloading?

Ans: Operator overloading is a function where different operators are applied and depends on the arguments. Operator,-,* can be used to pass through the function, and it has their own precedence to execute.

Example:

```
Class complex {
Double real,
Imag; public: complex (double r, double i) : real(r),
Imag (i) {} complex operator+ (complex a, complex b);
Complex operator*(complex a, complex b);
Complex& operator=(complex a, complex b);
}
```

A=1.2, b=6

Q16. What Is An Abstract Class?

Ans: An abstract class is a class which cannot be instantiated. Creation of an object is not possible with abstract class, but it can be inherited. An abstract class can contain only abstract method. Java allows only abstract method in abstract class while for other language it allows non-abstract method as well.

Q17. What is a ternary operator?

Ans: Ternary operator is said to be an operator which takes three arguments. Arguments and results are of different data types , and it is depends on the function. Ternary operator is also called as conditional operator.

Q18. What Is the use of Finalize Method?

Ans: Finalize method helps to perform cleanup operations on the resources which are not currently used. Finalize method is protected, and it is accessible only through this class or by a derived class.

Q19. What Are Different Types Of Arguments?

Ans: A parameter is a variable used during the declaration of the function or subroutine and arguments are passed to the function, and it should match with the parameter defined. There are two types of Arguments.

1-Call by Value –Value passed will get modified only inside the function, and it returns the same value whatever it is passed it into the function.

2-Call by Reference –Value passed will get modified in both inside and outside the functions and it returns the same or different value.

Q20. What Is Super Keyword?

Ans: Super keyword is used to invoke overridden method which overrides one of its superclass methods. This keyword allows to access overridden methods and also to access hidden members of the superclass. It also forwards a call from a constructor to a constructor in the superclass.

Q21. What Is Method Overriding?

Ans: Method overriding is a feature that allows sub class to provide implementation of a method that is already defined in the main class. This will overrides the implementation in the superclass by providing the same method name, same parameter and same return type.

Q22. What Is An Interface?

Ans: An interface is a collection of abstract method. If the class implements an inheritance, and then thereby inherits all the abstract methods of an interface.

Q23. What Is Exception Handling?

Ans: Exception is an event that occurs during the execution of a program. Exceptions can be of any type – Run time exception, Error exceptions. Those exceptions are handled properly through exception handling mechanism like try, catch and throw keywords.

Q24. What Are Tokens?

Ans: Token is recognized by a compiler and it cannot be broken down into component elements. Keywords, identifiers, constants, string literals and operators are examples of tokens. Even punctuation characters are also considered as tokens – Brackets, Commas, Braces and Parentheses.

Q25. Difference between overloading and overriding?

Ans: Overloading is static binding whereas Overriding is dynamic binding. Overloading is nothing but the same method with different arguments, and it may or may not return the same value in the same class itself. Overriding is the same method names with same arguments and returns types associates with the class and its child class.

Q26. Difference between class and an object?

Ans: An object is an instance of a class. Objects hold any information, but classes don't have any information. Definition of properties and functions can be done at class and can be used by the object. Class can have sub-classes, and an object doesn't have sub-objects.

Q27. What is an abstraction?

Ans: Abstraction is a good feature of OOPS, and it shows only the necessary details to the client of an object. Means, it shows only necessary details for an object, not the inner details of an object. Example – When you want to switch on television, it not necessary to show all the functions of TV. Whatever is required to switch on TV will be showed by using abstract class.

Q28. What Are Access Modifiers?

Ans: Access modifiers determine the scope of the method or variables that can be accessed from other various objects or classes. There are 5 types of access modifiers, and they are as follows:.

1-Private.

2-Protected.

3-Public.

4-Friend.

5-Protected Friend.

Q29. What is sealed modifiers?

Ans: Sealed modifiers are the access modifiers where it cannot be inherited by the methods. Sealed modifiers can also be applied to properties, events and methods. This modifier cannot be applied to static members.

Q30. How can we call the base method without creating an instance?

Ans: Yes, it is possible to call the base method without creating an instance. And that method should be,. Static method. Doing inheritance from that class.-Use Base Keyword from derived class.

Q31. What is the difference between new and override?

Answer: The new modifier instructs the compiler to use the new implementation instead of the base class function. Whereas, Override modifier helps to override the base class function.

Q32. What are the various types of constructors?

Ans: There are three various types of constructors, and they are as follows:

1-Default Constructor – With no parameters.

2-Parametric Constructor – With Parameters. Create a new instance of a class and also passing arguments simultaneously.

3-Copy Constructor – Which creates a new object as a copy of an existing object.

Q 33. What is early and late binding?

Answer: Early binding refers to assignment of values to variables during design time whereas late binding refers to assignment of values to variables during run time.

Q34. What is 'this' pointer?

Ans: THIS pointer refers to the current object of a class. THIS keyword is used as a pointer which differentiates between the current object with the global object. Basically, it refers to the current object.

Q35. What is the difference between structure and a class?

Ans: Structure default access type is public, but class access type is private. A structure is used for grouping data whereas class can be used for grouping data and methods. Structures are exclusively used for data and it doesn't require strict validation, but classes are used to encapsulates and inherit data which requires strict validation.

Q36. What is the default access modifier in a class?

Ans: The default access modifier of a class is Private by default.

Q37. What is pure virtual function?

Ans: A pure virtual function is a function which can be overridden in the derived class but cannot be defined. A virtual function can be declared as Pure by using the operator =0.

Example:

Virtual void function1 () // Virtual, Not pure

Virtual void function2 () = 0 //Pure virtual

Q38. What are all the operators that cannot be overloaded?

Ans: Following are the operators that cannot be overloaded -.

1-Scope Resolution (::)

2-Member Selection (.)

3-Member selection through a pointer to function (.*)

Q 39. What is dynamic or run time polymorphism?

Ans: Dynamic or Run time polymorphism is also known as method overriding in which call to an overridden function is resolved during run time, not at the compile time. It means having two or more methods with the same name, same signature but with different implementation.

Q40. Do we require parameter for constructors?

Ans: No, we do not require parameter for constructors.

Q 41. What is a copy constructor?

Ans: This is a special constructor for creating a new object as a copy of an existing object. There will be always only on copy constructor that can be either defined by the user or the system.

Q 42. What does the keyword virtual represented in the method definition?

Ans: It means, we can override the method.

Q43. What are base class, sub class and super class?

Ans: Base class is the most generalized class, and it is said to be a root class. Sub class is a class that inherits from one or more base classes. Super class is the parent class from which another class inherits.

Q 44. What is static and dynamic binding?

Ans: Binding is nothing but the association of a name with the class. Static binding is a binding in which name can be associated with the class during compilation time , and it is also called as early Binding. Dynamic binding is a binding in which name can be associated with the class during execution time , and it is also called as Late Binding.

Q45. How many instances can be created for an abstract class?

Ans: Zero instances will be created for an abstract class.

Q46. Which keyword can be used for overloading?

Ans: Operator keyword is used for overloading.

Q 47. What is the default access specifier in a class definition?

Ans: Private access specifier is used in a class definition.

Q48. Which oops concept is used as reuse mechanism?

Ans: Inheritance is the OOPS concept that can be used as reuse mechanism.

Q 49. Which oops concept exposes only necessary information to the calling functions?

Ans: Data Hiding / Abstraction

Q 50. What are the types of constructors?

Ans: Basically constructors are 5 types those are

1-Default Constructor

2-Parameterized Constructor

3-Copy Constructor

4-Static Constructor

5-Private Constructor

Q51.What is OOPS?

Ans: Object Oriented Programming System is the programming technique to write programs based on the real world objects. The states and behaviours of an object are represented as the member variables and methods. In OOPS programming programs are organized around objects and data rather than actions and logic.

Q52.What are the advantages of OOPS concepts

Ans: Major advantages of OOPS programming are;

1. Simplicity: OOPS programming objects model real world objects, so the complexity is reduced and the program structure is clear.

2-Modularity: Each object forms a separate entity whose internal workings are decoupled from other parts of the system.

3-Modifiability: It is easy to make minor changes in the data representation or the procedures in an OO program. Changes inside a class do not affect any other part of a program, since the only public interface that the external world has to a class is through the use of methods.

4-Extensibility: Adding new features or responding to changing operating environments can be solved by introducing a few new objects and modifying some existing ones.

5-Maintainability: Objects can be maintained separately, making locating and fixing problems easier.

6-Reusability: Objects can be reused in different programs.

Q53.What is the difference between Procedural programming and OOPS?

Ans: 1-Procedural language is based on functions but object oriented language is based on real world objects.

2-Procedural language gives importance on the sequence of function execution but object oriented language gives importance on states and behaviours of the objects.

3-Procedural language exposes the data to the entire program but object oriented language encapsulates the data.

4-Procedural language follows top down programming paradigm but object oriented language follows bottom up programming paradigm.

5-Procedural language is complex in nature so it is difficult to modify, extend and maintain but object oriented language is less complex in nature so it is easier to modify, extend and maintain.

6-Procedural language provides less scope of code reuse but object oriented language provides more scope of code reuse.

Q54.What are the core concepts of OOPS?

Ans: OOPS core concepts are;

1-Abstraction

2-Encapsulation

3-Polymorphism

4-Inheritance

5-Composition

6-Association

7-Aggregation

Q55.What is Abstraction?

Ans: Abstraction is an OOPS concept to construct the structure of the real world objects. During this construction only the general states and behaviours are taken and more specific states and behaviours are left aside for the implementers.

Q56.What is Encapsulation?

Ans: Encapsulation is an OOPS concept to create and define the permissions and restrictions of an object and its member variables and methods. A very simple example to explain the concept is to make the member variables of a class private and providing public getter and setter methods. Java provides four types of access level modifiers: public, protected, no modifier and private.

Q57.What is the difference between Abstraction and Encapsulation?

Ans:1-"Program to interfaces, not implementations" is the principle for Abstraction and "Encapsulate what varies" is the OO principle for Encapsulation.

2-Abstraction provides a general structure of a class and leaves the details for the implementers. Encapsulation is to create and define the permissions and restrictions of an object and its member variables and methods.

3-Abstraction is implemented in Java using interface and abstract class while Encapsulation is implemented using four types of access level modifiers: public, protected, no modifier and private.

Q58.What is Polymorphism?

Ans: Polymorphism is the occurrence of something in various forms. Java supports various forms of polymorphism like polymorphic reference variables, polymorphic method, polymorphic return types and polymorphic argument types.

Q59.What is Inheritance?

Ans: A subclass can inherit the states and behaviours of it's super class is known as inheritance.

Q60.What is multiple inheritance?

Ans: A child class inheriting states and behaviours from multiple parent classes is known as multiple inheritance.

Q61.What is the diamond problem in inheritance?

Ans: In case of multiple inheritance, suppose class A has two subclasses B and C, and a class D has two super classes B and C. If a method present in A is overridden by both B and C but not by D then from which class D will inherit that method B or C? This problem is known as diamond problem.

Q62.Why Java does not support multiple inheritance?

Ans: Java was designed to be a simple language and multiple inheritance introduces complexities like diamond problem. Inheriting states or behaviours from two different type of classes is a case which in reality very rare and it can be achieved easily through an object association.

Q63.What is Static Binding and Dynamic Binding?

Ans:1-Static or early binding is resolved at compile time. Method overloading is an example of static binding.

2-Dynamic or late or virtual binding is resolved at run time. Method overriding is an example of dynamic binding.

Q64.What is the meaning of "IS-A" and "HAS-A" relationship?

Ans: "IS-A" relationship implies inheritance. A sub class object is said to have "IS-A" relationship with the super class or interface. If class A extends B then A "IS-A" B. It is transitive, that is, if class A extends B and class B extends C then A "IS-A" C. The "instance of" operator in java determines the "IS-A" relationship.

When a class A has a member reference variable of type B then A "HAS-A" B. It is also known as Aggregation.

Q65.What is Association?

Ans: Association is a relationship between two objects with multiplicity.

Q66.What is Aggregation?

Ans: Aggregation is also known as "HAS-A" relationship. When class Car has a member reference variable of type Wheel then the relationship between the classes Car and Wheel is known as Aggregation. Aggregation can be understood as "whole to its parts" relationship.

Car is the whole and Wheel is part. Wheel can exist without the Car. Aggregation is a weak association.

Q67.What is Composition?

Ans: Composition is a special form of Aggregation where the part cannot exist without the whole. Composition is a strong Association. Composition relationship is represented like aggregation with one difference that the diamond shape is filled.

Q68.What is Dependency?

Ans: When one class depends on another because it uses that at some point in time then this relationship is known as Dependency. One class depends on another if the independent class is a parameter variable or local variable of a method of the dependent class. A Dependency is drawn as a dotted line from the dependent class to the independent class with an open arrowhead pointing to the independent class.

Q69.What is the difference between Association and Dependency?

Ans: The main difference between Association and Dependency is in case of Association one class has an attribute or member variable of the other class type but in case of Dependency a method takes an argument of the other class type or a method has a local variable of the other class type.

Q70.What is a Class?

Ans: A class is the specification or template of an object.

1-What is an Object?

2-Object is instance of class.

Q71.Define abstraction in software in general.

Ans: In computing, abstraction means separating concepts from details through a hierarchy of layers.

Examples:

1-Abit as an abstraction of a hardware electrical signal

2-Avariable name is an abstraction of a memory location

3-Adata type is an abstraction of a memory representation scheme by a compiler

4-Afunction call is an abstraction to the actual detailed code implemented in the body of the function

Q72.Define data abstraction.

Ans: 1-Creating a structure that defines a complex data type and legal operations on it. The design makes the interface of the data and operations accessible but hides the actual implementation of them.

2-Example: a List class can choose to implement the container of the data elements as an array or as a linked list. This container is kept hidden from the user interface but public interface is provided to element read/write operations.

Q73.Define process abstraction.

Ans: 1-Separating the process (method, procedure, function, subroutine or subprogram) signature/prototype from the detailed actual definition and implementation, so that the user is not concerned with how the process is implemented but rather how to use it.

2-Example: a method is defined to sort a list of elements and can be implemented using insertion sort or selection sort and the only thing the user has to worry about is how to call it.

Q74.What is encapsulation?

Ans: 1-It is a mechanism to package the data and its operations inside one structure

2-It is a data protection mechanism to dictate the way data is accessed (read/write)

Q75.What is information hiding?

Ans: The internal representation and implementation of an object is hidden from users. This concept encompasses: data abstraction, process abstraction and encapsulation.

Q76. Define polymorphism.

Ans: 1-A mechanism that works in hierarchical inheritance where a certain behavior when invoked can have different forms based on the called function of the derived class.

2-Example: a base class Animal defines "move" and "makeSound " functions ;a derived class Dog redefines the two functions to walk and bark respectively using the same function names in the base class, where as a Bird class redefines the two functions to fly and chirp.

3-Polymorphism requires that functions in base class to be defined as "Virtual" so derived classes can redefine them.

4-Virtual functions are bound dynamically at run-time whereas other functions are statically bound at compile time.

Q77. What are accessors?

Ans: 1-They are the functions in the public interface that provide accessibility to the private data members for "read" operations.

2-Example: see the getX, getY functions in the Point class.

Q78. What are mutators?

Ans: 1-They are the functions in the public interface that provide accessibility to the private data members for "write" operations.

2-Example: see the setX, setY functions in the Point class.

Q79.How to do inline functions work?

Ans: 1-In inline functions, there is no function calls, the compiler replace every instance of their calls in the code with the body of the function saving the overhead associated with the function call mechanism.

2-"inline" keyword can be used to specify a function as an inline.

Q80. What is public interface?

Ans: 1-Is a set of public functions defined in the "public:" section in the class body. This section provides the data accessibility of read/write operations as well as other service functions.

2-Example: see the "public" section in Point class. We can add another service function that prints the data members x and y to the stdout as follows:

```
void print()
{ cout << " X-coordinate = " << x << endl << Y-coordinate = " << y << endl; }
```

Q81.What are private functions?

Ans: 1-They are functions defined in the "private" section of the class body where they are only visible to the class itself; They are used when the class needs to some internal functions that the user do not need to know about and they are typically helper functions that help other public functions.

2-Example: suppose that in the Point3D function we need a function that will generate a random id# for the current object and we want to hide this function from the user so whenever a random ID is needed other functions in the class can automatically call this function:

```
void Point3D::randID(){
id = 1000 + rand() % (9000);
}
```

Q82.Define access modifiers.

Ans: 1-Private: the section in the class that is only visible to the class itself.

2-Protected: the section in the class that is visible to the class and to its derived classes.

3-Public: the section in the class that is visible to all: class, derived classes and outer world.

Q83. Define state, data member, attribute, property.

Ans: 1-They all refer to one thing, that is, the data fields that provide data specification for the class.

2-Example: see the data members defined in the "protected:" section below.

```
class Point
{
private:
protected:
double x;
double y;
int id;
```

```
double size;
void randID();
Public:
};
```

Q84. Define function, method & behavior.

Ans: They all refer to the same thing, the means of modelling how the object can operate and behave. While data members model states and attributes, functions model actions, operations and behaviors.

Q85. What is class specification?

Ans: The entire class body where every data member declarations, function members prototypes, constructors, accessors mutator prototypes and other elements of class declaration; all of these are typically kept in a header file ".h" apart from the implementation file of the class ".cpp".

Q86. What is class implementation?

Ans: It is typically kept in a separate file apart from class specification. In the implementation file, all functions declared in the class specification section/file are defined including the constructors, destructor, accessors, mutators and other service functions.

Q87. Write the simplest class possible that does nothing.class.

Ans: MyClass { };

Q88. What is an instance of a class?

Ans: 1-It is an object instantiated from a class. An instance of a class is created at runtime using one the available constructors. By setting its states/attributes/data members it becomes a specific instant/object of the class.

Q89. Define instantiation.

Ans: To create a new instance (object) of a class. Only when you instantiate an object, then you can use it and communicate with it.

Q90. What is a message?

Ans: Objects interact with each other using messaging which is simply calling each other's functions and receiving/causing the desired response.

Q91. Explain memory leakage.

Ans:1-Occurs when the programmer repeatedly allocates dynamic memory and does not release it when it is no longer needed, this will eventually exhaust all memory resources and cause a fatal error that will crash the program.

2-Example: in version one the function keeps allocating memory for about 10GB, if the system has that space available, it would work but with a lot of wasted space.

Version-2 correctly and safely releases the allocated memory after it has finished using it and will use 100MB at any given time.

Q92. What is exception handling?

Ans:1-Is a construct and a mechanism that manages the occurrence of errors/abnormalities at run time. Exception handling is designed to separate the error handling code from the original code to gracefully handle the fault situation by transferring the program control to the exception handling code. Many aspects of software development process can be improved such as: safety, readability, writeability, maintainability, etc.

2-Examples of exceptions: Divide by zero, sqrt of a negative, un initialized variable, exceeding array limits, dereferencing a dangling pointer, etc.

Q93. What is class extension?

Ans: It means inheritance; where a derived class inherits all of the base class content and extends the data member or/and the function members. In OOP inheritance implies extension where, there is no point of just inheriting the same content of the base class and not adding any data or function members, if that is the case then, the base class by itself could be used directly and no need for inheritance.

Q94. How to resolve conflicts in naming between multiple parent classes if they are called from a child?

Ans:1-Derived class redefines the multiply-defined function in parents or,

2-Derived class invokes member function in a particular base class using scope resolution operator ::

3-Compiler errors occur if derived class uses base class function without one of above solutions.

Q95. What is code reuse?

It is a mechanism in OOP that facilitates the utilization of previously created code; its main advantages include: time saving, better resource utilization and reducing redundancy. One good example of code reuse is inheritance.

Q96. Compare function overloading vs. overriding vs. Redefinition.

Ans:1-Overloading: defining multiple function using the same function name but with different parameter list ether in type or number. In classes, the overloaded constructor is an example where you can have many constructors with different parameter list signatures.

2-Redefinition: if a base class non-virtual function is redefined in a derived class, it is called redefined and it is statically bound.

3-Overriding: Redefinition: if a base class virtual function is redefined in a derived class, it is called overridden and it is dynamically bound.

Q97.What do you mean by 'void' return type?

Ans: All functions should return a value as per the general syntax. However, in case, if we don't want a function to return any value, we use "void" to indicate that. This means that we use "void" to indicate that the function has no return value or it returns "void".

Q98-Why are arrays usually processed with for loop?

Ans: Array uses the index to traverse each of its elements. If A is an array then each of its element is accessed as A[i]. Programmatically, all that is required for this to work is an iterative block with a loop variable i that serves as an index (counter) incrementing from 0 to A.length-1.

This is exactly what a loop does and this is the reason why we process arrays using for loops.

Q99- State the difference between delete and delete[].

Ans: "delete[]" is used to release the memory allocated to an array which was allocated using new[]. **"delete"** is used to release one chunk of memory which was allocated using new.

1) What is Java?

Java is the high-level, object-oriented, robust, secure programming language, platform-independent, high performance, Multithreaded, and portable programming language. It was developed by **James Gosling** in June 1991. It can also be known as the platform as it provides its own JRE and API.

---

2) What are the differences between C++ and Java?

The differences between C++ and Java are given in the following table.

Comparison Index	C++	Java
Platform-independent	C++ is platform-dependent.	Java is platform-independent.
Mainly used for	C++ is mainly used for system programming.	Java is mainly used for application programming. It is widely used in window, web-based, enterprise and mobile applications.
Design Goal	C++ was designed for systems and applications programming.	Java was designed and created as an interpreter for printing systems but

	It was an extension of C programming language.	later extended as a support network computing. It was designed with a goal of being easy to use and accessible to a broader audience.
Goto	C++ supports the goto statement.	Java doesn't support the goto statement.
Multiple inheritance	C++ supports multiple inheritance.	Java doesn't support multiple inheritance through class. It can be achieved by interfaces in java.
Operator Overloading	C++ supports operator overloading.	Java doesn't support operator overloading.
Pointers	C++ supports pointers. You can write pointer program in C++.	Java supports pointer internally. However, you can't write the pointer program in java. It means java has restricted pointer support in Java.

Compiler and Interpreter	C++ uses compiler only. C++ is compiled and run using the compiler which converts source code into machine code so, C++ is platform dependent.	Java uses compiler and interpreter both. Java source code is converted into bytecode at compilation time. The interpreter executes this bytecode at runtime and produces output. Java is interpreted that is why it is platform independent.
Call by Value and Call by reference	C++ supports both call by value and call by reference.	Java supports call by value only. There is no call by reference in java.
Structure and Union	C++ supports structures and unions.	Java doesn't support structures and unions.
Thread Support	C++ doesn't have built-in support for threads. It relies on third-party libraries for thread support.	Java has built-in thread support.

Documentation comment	C++ doesn't support documentation comment.	Java supports documentation comment (/** ... */) to create documentation for java source code.
Virtual Keyword	C++ supports virtual keyword so that we can decide whether or not override a function.	Java has no virtual keyword. We can override all non-static methods by default. In other words, non-static methods are virtual by default.
unsigned right shift >>>	C++ doesn't support >>> operator.	Java supports unsigned right shift >>> operator that fills zero at the top for the negative numbers. For positive numbers, it works same like >> operator.
Inheritance Tree	C++ creates a new inheritance tree always.	Java uses a single inheritance tree always because all classes are the child of Object class in java. The object class is the root of the inheritance tree in java.

Hardware	C++ is nearer to hardware.	Java is not so interactive with hardware.
Object-oriented	C++ is an object-oriented language. However, in C language, single root hierarchy is not possible.	Java is also an object-oriented language. However, everything (except fundamental types) is an object in Java. It is a single root hierarchy as everything gets derived from java.lang.Object.

---

3) List the features of Java Programming language.

There are the following features in Java Programming Language.

- **Simple:** Java is easy to learn. The syntax of Java is based on C++ which makes easier to write the program in it.

- **Object-Oriented:** Java follows the object-oriented paradigm which allows us to maintain our code as the combination of different type of objects that incorporates both data and behavior.

- **Portable:** Java supports read-once-write-anywhere approach. We can execute the Java program on every machine. Java program (.java) is converted to bytecode (.class) which can be easily run on every machine.

- **Platform Independent:** Java is a platform independent programming language. It is different from other programming languages like C and C++ which needs a platform to be executed. Java comes with its platform on which its code is executed. Java doesn't depend upon the operating system to be executed.

- **Secured:** Java is secured because it doesn't use explicit pointers. Java also provides the concept of ByteCode and Exception handling which makes it more secured.

- **Robust:** Java is a strong programming language as it uses strong memory management. The concepts like Automatic garbage collection, Exception handling, etc. make it more robust.

- **Architecture Neutral:** Java is architectural neutral as it is not dependent on the architecture. In C, the size of data types may vary according to the architecture (32 bit or 64 bit) which doesn't exist in Java.

- **Interpreted:** Java uses the Just-in-time (JIT) interpreter along with the compiler for the program execution.

- **High Performance:** Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code. It is still a little bit slower than a compiled language (e.g., C++).

- **Multithreaded:** We can write Java programs that deal with many tasks at once by defining multiple threads. The main advantage of multi-threading is that it doesn't occupy memory for each thread. It shares a common memory area. Threads are important for multi-media, Web applications, etc.

- **Distributed:** Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications. This feature of Java makes us able to access files by calling the methods from any machine on the internet.

- **Dynamic:** Java is a dynamic language. It supports dynamic loading of classes. It means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

---

4) What do you understand by Java virtual machine?

Java Virtual Machine is a virtual machine that enables the computer to run the Java program. JVM acts like a run-time engine which calls the main method present in the Java code. JVM is the specification which must be implemented in the computer system. The Java code is compiled by JVM to be a Bytecode which is machine independent and close to the native code.

---

5) What is the difference between JDK, JRE, and JVM?

JVM

JVM is an acronym for Java Virtual Machine; it is an abstract machine which provides the runtime environment in which Java bytecode can be executed. It is a specification which specifies the working of Java Virtual Machine. Its implementation has been provided by Oracle and other companies. Its implementation is known as JRE.

JVMs are available for many hardware and software platforms (so JVM is platform dependent). It is a runtime instance which is created when we run the Java class. There are three notions of the JVM: specification, implementation, and instance.

JRE

JRE stands for Java Runtime Environment. It is the implementation of JVM. The Java Runtime Environment is a set of software tools which are used for developing Java applications. It is used to provide the runtime environment. It is the implementation of JVM. It physically exists. It contains a set of libraries + other files that JVM uses at runtime.

JDK

JDK is an acronym for Java Development Kit. It is a software development environment which is used to develop Java applications and applets. It physically exists. It contains JRE + development tools. JDK is an implementation of any one of the below given Java Platforms released by Oracle Corporation:

- Standard Edition Java Platform

- Enterprise Edition Java Platform
- Micro Edition Java Platform

More Details.

---

6) How many types of memory areas are allocated by JVM?

Many types:

1. **Class(Method) Area:** Class Area stores per-class structures such as the runtime constant pool, field, method data, and the code for methods.
2. **Heap:** It is the runtime data area in which the memory is allocated to the objects
3. **Stack:** Java Stack stores frames. It holds local variables and partial results, and plays a part in method invocation and return. Each thread has a private JVM stack, created at the same time as the thread. A new frame is created each time a method is invoked. A frame is destroyed when its method invocation completes.
4. **Program Counter Register:** PC (program counter) register contains the address of the Java virtual machine instruction currently being executed.
5. **Native Method Stack:** It contains all the native methods used in the application.

More Details.

---

7) What is JIT compiler?

Just-In-Time(JIT) compiler: It is used to improve the performance. JIT compiles parts of the bytecode that have similar functionality at the same time, and hence reduces the amount of time needed for compilation. Here the term “compiler” refers to a translator from the instruction set of a Java virtual machine (JVM) to the instruction set of a specific CPU.

---

8) What is the platform?

A platform is the hardware or software environment in which a piece of software is executed. There are two types of platforms, software-based and hardware-based. Java provides the software-based platform.

---

9) What are the main differences between the Java platform and other platforms?

There are the following differences between the Java platform and other platforms.

- Java is the software-based platform whereas other platforms may be the hardware platforms or software-based platforms.

- Java is executed on the top of other hardware platforms whereas other platforms can only have the hardware components.

---

10) What gives Java its 'write once and run anywhere' nature?

The bytecode. Java compiler converts the Java programs into the class file (Byte Code) which is the intermediate language between source code and machine code. This bytecode is not platform specific and can be executed on any computer.

---

11) What is classloader?

Classloader is a subsystem of JVM which is used to load class files. Whenever we run the java program, it is loaded first by the classloader. There are three built-in classloaders in Java.

1. **Bootstrap ClassLoader**: This is the first classloader which is the superclass of Extension classloader. It loads the *rt.jar* file which contains all class files of Java Standard Edition like java.lang package classes, java.net package classes, java.util package classes, java.io package classes, java.sql package classes, etc.
2. **Extension ClassLoader**: This is the child classloader of Bootstrap and parent classloader of System classloader. It loads the jar files located inside *$JAVA_HOME/jre/lib/ext* directory.

3. **System/Application ClassLoader**: This is the child classloader of Extension classloader. It loads the class files from the classpath. By default, the classpath is set to the current directory. You can change the classpath using "-cp" or "-classpath" switch. It is also known as Application classloader.

---

12) Is Empty .java file name a valid source file name?

Yes, Java allows to save our java file by **.java** only, we need to compile it by **javac .java** and run by **java classname** Let's take a simple example:

```
1. //save by .java only
2. class A{
3. public static void main(String args[]){
4. System.out.println("Hello java");
5. }
6. }
7. //compile by javac .java
8. //run by     java A
```

compile it by **javac .java**

run it by **java A**

13) Is delete, next, main, exit or null keyword in java?

No.

---

14) If I don't provide any arguments on the command line, then what will the value stored in the String array passed into the main() method, empty or NULL?

It is empty, but not null.

---

15) What if I write static public void instead of public static void?

The program compiles and runs correctly because the order of specifiers doesn't matter in Java.

---

16) What is the default value of the local variables?

The local variables are not initialized to any default value, neither primitives nor object references.

---

17) What are the various access specifiers in Java?

In Java, access specifiers are the keywords which are used to define the access scope of the method, class, or a variable. In Java, there are four access specifiers given below.

- **Public** The classes, methods, or variables which are defined as public, can be accessed by any class or method.
- **Protected** Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.
- **Default** Default are accessible within the package only. By default, all the classes, methods, and variables are of default scope.
- **Private** The private class, methods, or variables defined as private can be accessed within the class only.

---

18) What is the purpose of static methods and variables?

The methods or variables defined as static are shared among all the objects of the class. The static is the part of the class and not of the object. The static variables are stored in the class area, and we do not need to create the object to access such variables. Therefore, static is used in the case, where we need to define variables or methods which are common to all the objects of the class.

For example, In the class simulating the collection of the students in a college, the name of the college is the common attribute to all the students. Therefore, the college name will be defined as **static**.

---

19) What are the advantages of Packages in Java?

There are various advantages of defining packages in Java.

- Packages avoid the name clashes.
- The Package provides easier access control.
- We can also have the hidden classes that are not visible outside and used by the package.
- It is easier to locate the related classes.

---

20) What is the output of the following Java program?

```
1. class Test
2. {
3.     public static void main (String args[])
4.     {
5.         System.out.println(10 + 20 + "Javatpoint");
```

```
6.         System.out.println("Javatpoint" + 10 + 20);
7.     }
8. }
```

The output of the above code will be

30Javatpoint

Javatpoint1020

Explanation

In the first case, 10 and 20 are treated as numbers and added to be 30. Now, their sum 30 is treated as the string and concatenated with the string **Javatpoint**. Therefore, the output will be **30Javatpoint**.

In the second case, the string Javatpoint is concatenated with 10 to be the string **Javatpoint10** which will then be concatenated with 20 to be **Javatpoint1020**.

---

21) What is the output of the following Java program?

```
1. class Test
2. {
```

```
3.    public static void main (String args[])
4.    {
5.        System.out.println(10 * 20 + "Javatpoint");
6.        System.out.println("Javatpoint" + 10 * 20);
7.    }
8. }
```

The output of the above code will be

200Javatpoint

Javatpoint200

Explanation

In the first case, The numbers 10 and 20 will be multiplied first and then the result 200 is treated as the string and concatenated with the string **Javatpoint** to produce the output **200Javatpoint**.

In the second case, The numbers 10 and 20 will be multiplied first to be 200 because the precedence of the multiplication is higher than addition. The result 200 will be treated as the string and concatenated with the string **Javatpoint**to produce the output as **Javatpoint200**.

---

22) What is the output of the following Java program?

```
1. class Test
2. {
3.     public static void main (String args[])
4.     {
5.         for(int i=0; 0; i++)
6.         {
7.             System.out.println("Hello Javatpoint");
8.         }
9.     }
10.     }
```

The above code will give the compile-time error because the for loop demands a boolean value in the second part and we are providing an integer value, i.e., 0.

Core Java - OOPs Concepts: Initial OOPs Interview Questions

There is given more than 50 OOPs (Object-Oriented Programming and System) interview questions. However, they have been categorized in many sections such as constructor interview questions, static interview questions, Inheritance Interview questions, Abstraction interview question, Polymorphism interview questions, etc. for better understanding.

23) What is object-oriented paradigm?

It is a programming paradigm based on objects having data and methods defined in the class to which it belongs. Object-oriented paradigm aims to incorporate the advantages of modularity and reusability. Objects are the instances of classes which interacts with one another to design applications and programs. There are the following features of the object-oriented paradigm.

- Follows the bottom-up approach in program design.
- Focus on data with methods to operate upon the object's data
- Includes the concept like Encapsulation and abstraction which hides the complexities from the user and show only functionality.
- Implements the real-time approach like inheritance, abstraction, etc.
- The examples of the object-oriented paradigm are C++, Simula, Smalltalk, Python, C#, etc.

---

24) What is an object?

The Object is the real-time entity having some state and behavior. In Java, Object is an instance of the class having the instance variables as the state of the object and the methods as the behavior of the object. The object of a class can be created by using the **new** keyword.

25) What is the difference between an object-oriented programming language and object-based programming language?

There are the following basic differences between the object-oriented language and object-based language.

- Object-oriented languages follow all the concepts of OOPs whereas, the object-based language doesn't follow all the concepts of OOPs like inheritance and polymorphism.
- Object-oriented languages do not have the inbuilt objects whereas Object-based languages have the inbuilt objects, for example, JavaScript has window object.
- Examples of object-oriented programming are Java, C#, Smalltalk, etc. whereas the examples of object-based languages are JavaScript, VBScript, etc.

---

26) What will be the initial value of an object reference which is defined as an instance variable?

All object references are initialized to null in Java.

Core Java - OOPs Concepts: Constructor Interview Questions

---

27) What is the constructor?

The constructor can be defined as the special type of method that is used to initialize the state of an object. It is invoked when the class is instantiated, and the memory is allocated for the object. Every time, an object is created using the **new** keyword, the default constructor of the class is called. The name of the constructor must be similar to the class name. The constructor must not have an explicit return type.

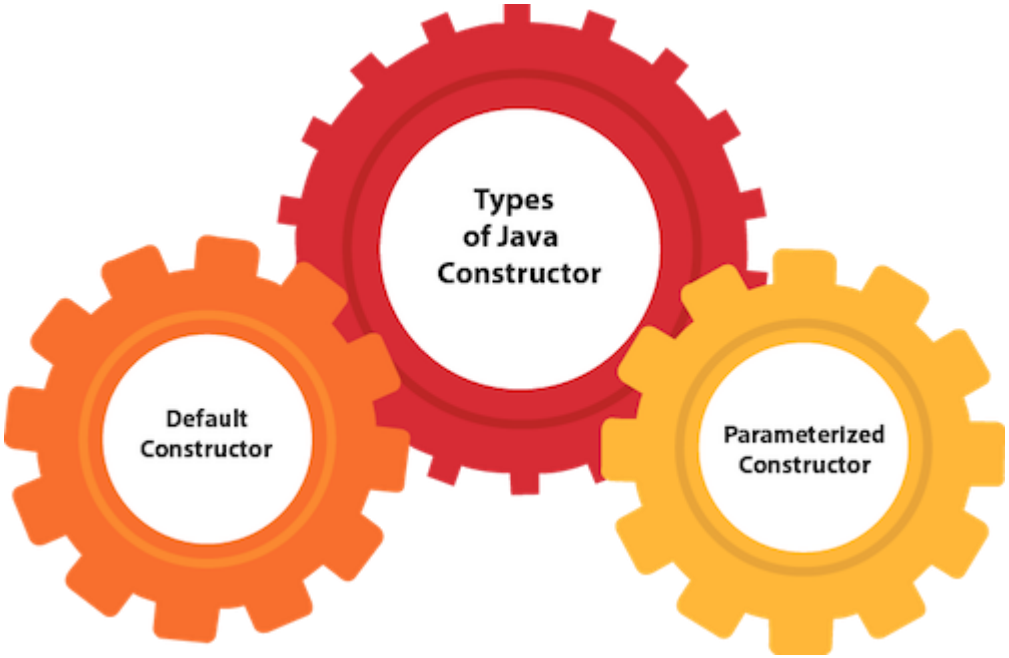
More Details.

---

28) How many types of constructors are used in Java?

Based on the parameters passed in the constructors, there are two types of constructors in Java.

- **Default Constructor:** default constructor is the one which does not accept any value. The default constructor is mainly used to initialize the instance variable with the default values. It can also be used for performing some useful task on object creation. A default constructor is invoked implicitly by the compiler if there is no constructor defined in the class.
- **Parameterized Constructor:** The parameterized constructor is the one which can initialize the instance variables with the given values. In other words, we can say that the constructors which can accept the arguments are called parameterized constructors.



---

29) What is the purpose of a default constructor?

The purpose of the default constructor is to assign the default value to the objects. The java compiler creates a default constructor implicitly if there is no constructor in the class.

```
1. class Student3{
2. int id;
3. String name;
4.
5. void display(){System.out.println(id+" "+name);}
```

```
6.
7. public static void main(String args[]){
8. Student3 s1=new Student3();
9. Student3 s2=new Student3();
10.     s1.display();
11.     s2.display();
12.     }
13.     }
```

Test it Now

Output:

```
0 null
0 null
```

Explanation: In the above class, you are not creating any constructor, so compiler provides you a default constructor. Here 0 and null values are provided by default constructor.

More Details.

---

30) Does constructor return any value?

Ans: yes, The constructor implicitly returns the current instance of the class (You can't use an explicit return type with the constructor). More Details.

---

31)Is constructor inherited?

No, The constructor is not inherited.

---

32) Can you make a constructor final?

No, the constructor can't be final.

---

33) Can we overload the constructors?

Yes, the constructors can be overloaded by changing the number of arguments accepted by the constructor or by changing the data type of the parameters. Consider the following example.

```
1. class Test
2. {
3.     int i;
4.     public Test(int k)
5.     {
6.         i=k;
7.     }
8.     public Test(int k, int m)
9.     {
10.            System.out.println("Hi I am assigning the value max(k, m) to i");
11.            if(k>m)
12.            {
```

```
13.             i=k;
14.         }
15.         else
16.         {
17.             i=m;
18.         }
19.     }
20. }
21. public class Main
22. {
23.     public static void main (String args[])
24.     {
25.         Test test1 = new Test(10);
26.         Test test2 = new Test(12, 15);
27.         System.out.println(test1.i);
28.         System.out.println(test2.i);
29.     }
30. }
31.
```

In the above program, The constructor Test is overloaded with another constructor. In the first call to the constructor, The constructor with one argument is called, and i will be initialized with the value 10. However, In the second call to the constructor, The constructor with the 2 arguments is called, and i will be initialized with the value 15.

---

34) What do you understand by copy constructor in Java?

There is no copy constructor in java. However, we can copy the values from one object to another like copy constructor in C++.

There are many ways to copy the values of one object into another in java. They are:

- By constructor
- By assigning the values of one object into another
- By clone() method of Object class

In this example, we are going to copy the values of one object into another using java constructor.

```
1. //Java program to initialize the values from one object to another
2. class Student6{
3.     int id;
4.     String name;
```

```
5.    //constructor to initialize integer and string
6.    Student6(int i,String n){
7.    id = i;
8.    name = n;
9.    }
10.         //constructor to initialize another object
11.         Student6(Student6 s){
12.         id = s.id;
13.         name =s.name;
14.         }
15.         void display(){System.out.println(id+" "+name);}
16.
17.         public static void main(String args[]){
18.         Student6 s1 = new Student6(111,"Karan");
19.         Student6 s2 = new Student6(s1);
20.         s1.display();
21.         s2.display();
22.        }
23.     }
```

Test it Now

Output:

```
111 Karan
111 Karan
```

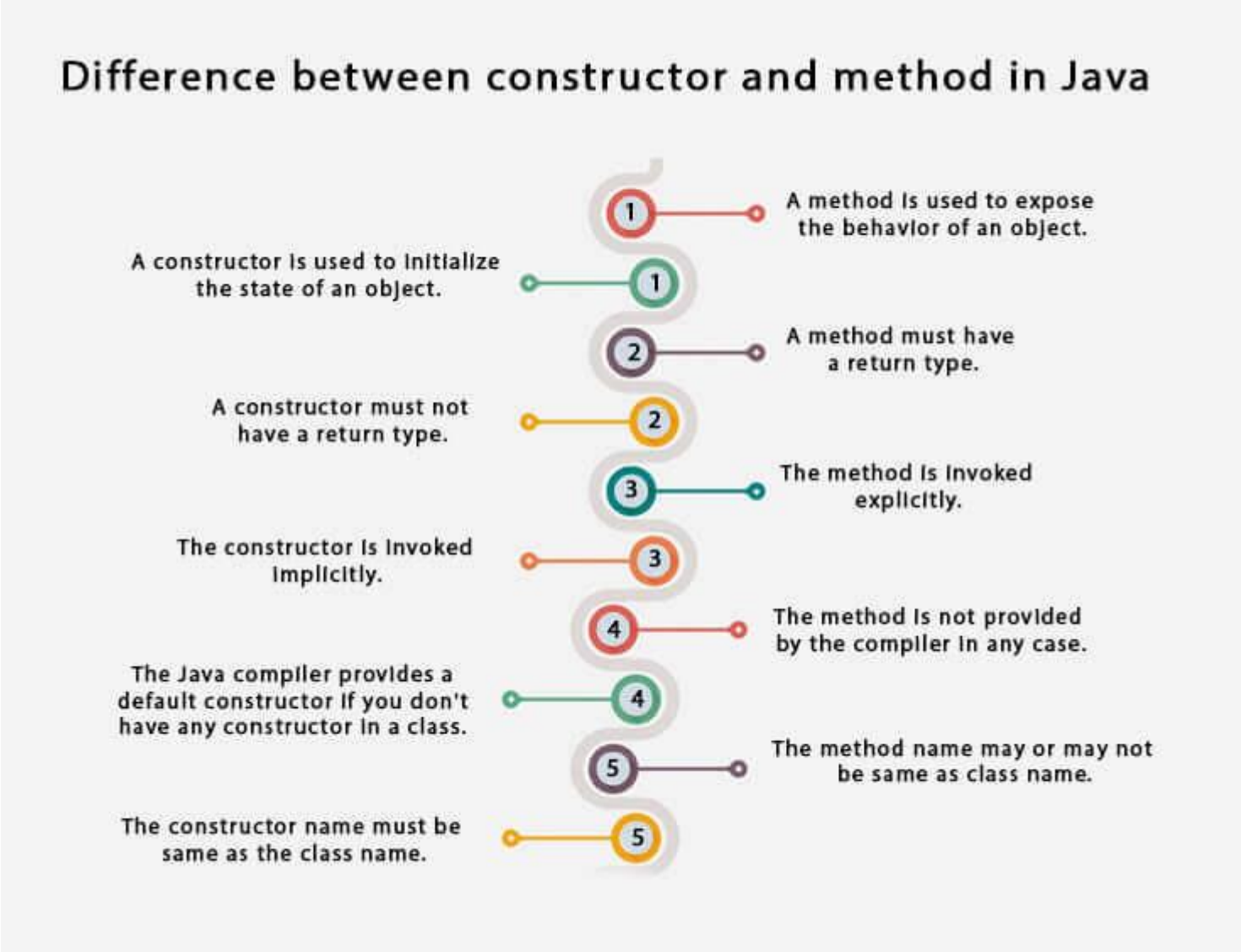
---

35) What are the differences between the constructors and methods?

There are many differences between constructors and methods. They are given below.

Java Constructor	**Java Method**
A constructor is used to initialize the state of an object.	A method is used to expose the behavior of an object.
A constructor must not have a return type.	A method must have a return type.
The constructor is invoked implicitly.	The method is invoked explicitly.
The Java compiler provides a default constructor if you don't have any constructor in a class.	The method is not provided by the compiler in any case.

The constructor name must be same as the class name.	The method name may or may not be same as class name.



36) What is the output of the following Java program?

```
1. public class Test
2. {
3.     Test(int a, int b)
4.     {
5.         System.out.println("a = "+a+" b = "+b);
6.     }
7.     Test(int a, float b)
8.     {
9.         System.out.println("a = "+a+" b = "+b);
10.        }
11.        public static void main (String args[])
12.        {
13.            byte a = 10;
14.            byte b = 15;
15.            Test test = new Test(a,b);
16.        }
17.    }
```

The output of the following program is:

```
a = 10 b = 15
```

Here, the data type of the variables a and b, i.e., byte gets promoted to int, and the first parameterized constructor with the two integer parameters is called.

---

37) What is the output of the following Java program?

```
1. class Test
2. {
3.     int i;
4. }
5. public class Main
6. {
7.     public static void main (String args[])
8.     {
9.         Test test = new Test();
10.             System.out.println(test.i);
11.         }
12.     }
```

The output of the program is 0 because the variable i is initialized to 0 internally. As we know that a default constructor is invoked implicitly if there is no constructor in the class, the variable i is initialized to 0 since there is no constructor in the class.

---

38) What is the output of the following Java program?

```
1. class Test
2. {
3.     int test_a, test_b;
4.     Test(int a, int b)
5.     {
6.     test_a = a;
7.     test_b = b;
8.     }
9.     public static void main (String args[])
10.         {
11.             Test test = new Test();
12.             System.out.println(test.test_a+" "+test.test_b);
13.         }
14.     }
```

There is a **compiler error** in the program because there is a call to the default constructor in the main method which is not present in the class. However, there is only one parameterized constructor in the class Test. Therefore, no default constructor is invoked by the constructor implicitly.

Core Java - OOPs Concepts: static keyword Interview Questions

---

39) What is the static variable?

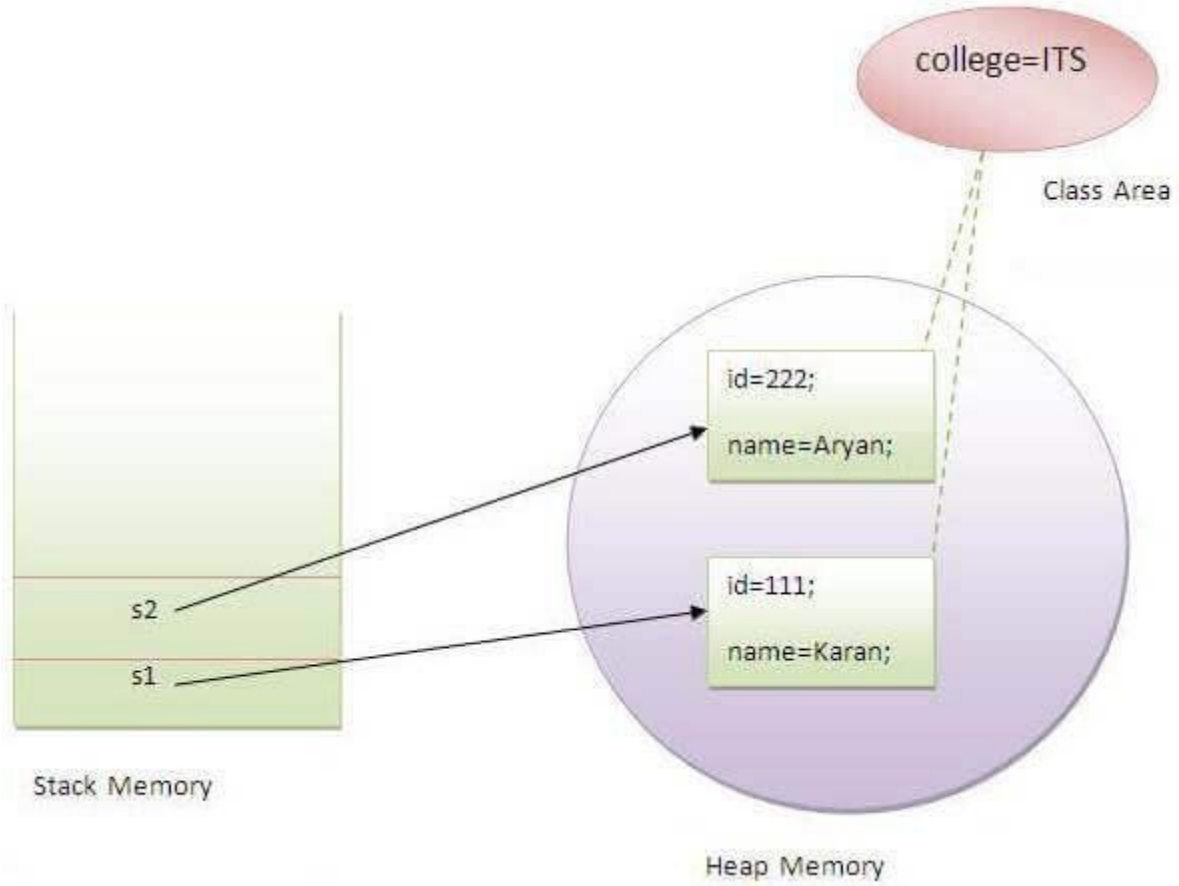
The static variable is used to refer to the common property of all objects (that is not unique for each object), e.g., The company name of employees, college name of students, etc. Static variable gets memory only once in the class area at the time of class loading. Using a static variable makes your program more memory efficient (it saves memory). Static variable belongs to the class rather than the object.

```
1. //Program of static variable
2.
3. class Student8{
4.    int rollno;
5.    String name;
6.    static String college ="ITS";
```

```
7.
8.    Student8(int r,String n){
9.    rollno = r;
10.         name = n;
11.         }
12.       void display (){System.out.println(rollno+" "+name+" "+college);}
13.
14.       public static void main(String args[]){
15.       Student8 s1 = new Student8(111,"Karan");
16.       Student8 s2 = new Student8(222,"Aryan");
17.
18.       s1.display();
19.       s2.display();
20.       }
21.      }
```

Test it Now

```
Output:111 Karan ITS
       222 Aryan ITS
```



More Details.

---

40) What is the static method?

- A static method belongs to the class rather than the object.
- There is no need to create the object to call the static methods.
- A static method can access and change the value of the static variable.

More Details.

---

41) What are the restrictions that are applied to the Java static methods?

Two main restrictions are applied to the static methods.

- The static method can not use non-static data member or call the non-static method directly.
- this and super cannot be used in static context as they are non-static.

---

42) Why is the main method static?

Because the object is not required to call the static method. If we make the main method non-static, JVM will have to create its object first and then call main() method which will lead to the extra memory allocation. More Details.

---

43) Can we override the static methods?

No, we can't override static methods.

44) What is the static block?

Static block is used to initialize the static data member. It is executed before the main method, at the time of classloading.

```
1. class A2{
2.   static{System.out.println("static block is invoked");}
3.   public static void main(String args[]){
4.    System.out.println("Hello main");
5.   }
6. }
```

Test it Now

```
Output: static block is invoked
        Hello main
```

More Details.

---

45) Can we execute a program without main() method?

Ans) Yes, one of the ways to execute the program without the main method is using static block. More Details.

---

46) What if the static modifier is removed from the signature of the main method?

Program compiles. However, at runtime, It throws an error "NoSuchMethodError."

---

47) What is the difference between static (class) method and instance method?

static or class method	**instance method**
1)A method that is declared as static is known as the static method.	A method that is not declared as static is known as the instance method.
2)We don't need to create the objects to call the static methods.	The object is required to call the instance methods.
3)Non-static (instance) members cannot be accessed in the static context (static method, static block, and static nested class) directly.	Static and non-static variables both can be accessed in instance methods.

4)For example: public static int cube(int n){ return n*n*n;}	For example: public void msg(){...}.

---

48) Can we make constructors static?

As we know that the static context (method, block, or variable) belongs to the class, not the object. Since Constructors are invoked only when the object is created, there is no sense to make the constructors static. However, if you try to do so, the compiler will show the compiler error.

---

49) Can we make the abstract methods static in Java?

In Java, if we make the abstract methods static, It will become the part of the class, and we can directly call it which is unnecessary. Calling an undefined method is completely useless therefore it is not allowed.

---

50) Can we declare the static variables and methods in an abstract class?

Yes, we can declare static variables and methods in an abstract method. As we know that there is no requirement to make the object to access the static context, therefore, we can access the static

context declared inside the abstract class by using the name of the abstract class. Consider the following example.

```
1. abstract class Test
2. {
3.     static int i = 102;
4.     static void TestMethod()
5.     {
6.         System.out.println("hi !! I am good !!");
7.     }
8. }
9. public class TestClass extends Test
10.     {
11.         public static void main (String args[])
12.         {
13.             Test.TestMethod();
14.             System.out.println("i = "+Test.i);
15.         }
16.     }
```

Output

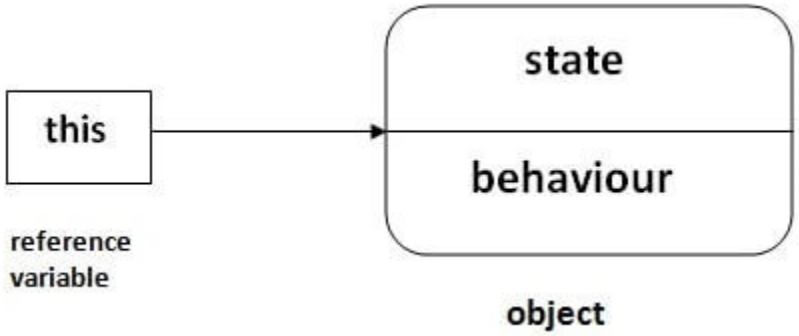
```
hi !! I am good !!
i = 102
```

Core Java - OOPs Concepts: Inheritance Interview Questions

---

51) What is **this** keyword in java?

The **this** keyword is a reference variable that refers to the current object. There are the various uses of this keyword in Java. It can be used to refer to current class properties such as instance methods, variable, constructors, etc. It can also be passed as an argument into the methods or constructors. It can also be returned from the method as the current class instance.



More Details.

---

52) What are the main uses of this keyword?

There are the following uses of **this** keyword.

- **this** can be used to refer to the current class instance variable.
- **this** can be used to invoke current class method (implicitly)
- **this()** can be used to invoke the current class constructor.
- **this** can be passed as an argument in the method call.
-
- **this** can be passed as an argument in the constructor call.
- **this** can be used to return the current class instance from the method.

---

53) Can we assign the reference to **this** variable?

No, this cannot be assigned to any value because it always points to the current class object and this is the final reference in Java. However, if we try to do so, the compiler error will be shown. Consider the following example.

```
1. public class Test
2. {
3.     public Test()
```

```
4.    {
5.        this = null;
6.        System.out.println("Test class constructor called");
7.    }
8.    public static void main (String args[])
9.    {
10.            Test t = new Test();
11.        }
12.    }
```

Output

```
Test.java:5: error: cannot assign a value to final variable this
        this = null;
        ^
1 error
```

---

54) Can **this** keyword be used to refer static members?

Yes, It is possible to use this keyword to refer static members because this is just a reference variable which refers to the current class object. However, as we know that, it is unnecessary to

access static variables through objects, therefore, it is not the best practice to use this to refer static members. Consider the following example.

```
1. public class Test
2. {
3.     static int i = 10;
4.     public Test ()
5.     {
6.         System.out.println(this.i);
7.     }
8.     public static void main (String args[])
9.     {
10.             Test t = new Test();
11.         }
12.     }
```

Output

```
10
```

---

55) How can constructor chaining be done using this keyword?

Constructor chaining enables us to call one constructor from another constructor of the class with respect to the current class object. We can use this keyword to perform constructor chaining within the same class. Consider the following example which illustrates how can we use this keyword to achieve constructor chaining.

```
1. public class Employee
2. {
3.     int id,age;
4.     String name, address;
5.     public Employee (int age)
6.     {
7.         this.age = age;
8.     }
9.     public Employee(int id, int age)
10.         {
11.             this(age);
12.             this.id = id;
13.         }
14.         public Employee(int id, int age, String name, String address)
15.         {
16.             this(id, age);
```

```
17.             this.name = name;
18.             this.address = address;
19.         }
20.         public static void main (String args[])
21.         {
22.             Employee emp = new Employee(105, 22, "Vikas", "Delhi");
23.             System.out.println("ID: "+emp.id+" Name:"+emp.name+" age:"+emp.age+" address: "+
    emp.address);
24.         }
25.
26.     }
```

Output

```
ID: 105 Name:Vikas age:22 address: Delhi
```

---

56) What are the advantages of passing this into a method instead of the current class object itself?

As we know, that this refers to the current class object, therefore, it must be similar to the current class object. However, there can be two main advantages of passing this into a method instead of the current class object.

- this is a final variable. Therefore, this cannot be assigned to any new value whereas the current class object might not be final and can be changed.
- this can be used in the synchronized block.

---

57) What is the Inheritance?

Inheritance is a mechanism by which one object acquires all the properties and behavior of another object of another class. It is used for Code Reusability and Method Overriding. The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also. Inheritance represents the IS-A relationship which is also known as a parent-child relationship.

There are five types of inheritance in Java.

- Single-level inheritance
- Multi-level inheritance
- Multiple Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

Multiple inheritance is not supported in Java through class.

More Details.

---

58) Why is Inheritance used in Java?

There are various advantages of using inheritance in Java that is given below.

- Inheritance provides code reusability. The derived class does not need to redefine the method of base class unless it needs to provide the specific implementation of the method.
- Runtime polymorphism cannot be achieved without using inheritance.
- We can simulate the inheritance of classes with the real-time objects which makes OOPs more realistic.
- Inheritance provides data hiding. The base class can hide some data from the derived class by making it private.
- Method overriding cannot be achieved without inheritance. By method overriding, we can give a specific implementation of some basic method contained by the base class.

---

59) Which class is the superclass for all the classes?

The object class is the superclass of all other classes in Java.

---

60) Why is multiple inheritance not supported in java?

To reduce the complexity and simplify the language, multiple inheritance is not supported in java. Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Since the compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have the same method or different, there will be a compile time error.

```
1. class A{
2. void msg(){System.out.println("Hello");}
3. }
4. class B{
5. void msg(){System.out.println("Welcome");}
6. }
7. class C extends A,B{//suppose if it were
8.
```

```
9.  Public Static void main(String args[]){
10.         C obj=new C();
11.         obj.msg();//Now which msg() method would be invoked?
12.     }
13.     }
```

Test it Now

```
Compile Time Error
```

---

61) What is aggregation?

Aggregation can be defined as the relationship between two classes where the aggregate class contains a reference to the class it owns. Aggregation is best described as a **has-a** relationship. For example, The aggregate class Employee having various fields such as age, name, and salary also contains an object of Address class having various fields such as Address-Line 1, City, State, and pin-code. In other words, we can say that Employee (class) has an object of Address class. Consider the following example.

Address.java

```
1. public class Address {
2. String city,state,country;
```

```
3.
4. public Address(String city, String state, String country) {
5.     this.city = city;
6.     this.state = state;
7.     this.country = country;
8. }
9.
10.      }
```

Employee.java

```
1. public class Emp {
2. int id;
3. String name;
4. Address address;
5.
6. public Emp(int id, String name,Address address) {
7.     this.id = id;
8.     this.name = name;
9.     this.address=address;
10.      }
11.
```

```
12.     void display(){
13.     System.out.println(id+" "+name);
14.     System.out.println(address.city+" "+address.state+" "+address.country);
15.     }
16.
17.     public static void main(String[] args) {
18.     Address address1=new Address("gzb","UP","india");
19.     Address address2=new Address("gno","UP","india");
20.
21.     Emp e=new Emp(111,"varun",address1);
22.     Emp e2=new Emp(112,"arun",address2);
23.
24.     e.display();
25.     e2.display();
26.
27.     }
28.     }
```

Output

```
111 varun
gzb UP india
```

```
112 arun
gno UP india
```

---

62) What is composition?

Holding the reference of a class within some other class is known as composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition. In other words, we can say that composition is the particular case of aggregation which represents a stronger relationship between two objects. Example: A class contains students. A student cannot exist without a class. There exists composition between class and students.

---

63) What is the difference between aggregation and composition?

Aggregation represents the weak relationship whereas composition represents the strong relationship. For example, the bike has an indicator (aggregation), but the bike has an engine (composition).

---

64) Why does Java not support pointers?

The pointer is a variable that refers to the memory address. They are not used in Java because they are unsafe(unsecured) and complex to understand.

---

65) What is super in java?

The **super** keyword in Java is a reference variable that is used to refer to the immediate parent class object. Whenever you create the instance of the subclass, an instance of the parent class is created implicitly which is referred by super reference variable. The super() is called in the class constructor implicitly by the compiler if there is no super or this.

```
1. class Animal{
2. Animal(){System.out.println("animal is created");}
3. }
4. class Dog extends Animal{
5. Dog(){
6. System.out.println("dog is created");
7. }
8. }
9. class TestSuper4{
10.     public static void main(String args[]){
11.     Dog d=new Dog();
```

```
12.     }
13.     }
```

Test it Now

Output:

```
animal is created
dog is created
```

More Details.

---

66) How can constructor chaining be done by using the super keyword?

```
1. class Person
2. {
3.     String name,address;
4.     int age;
5.     public Person(int age, String name, String address)
6.     {
7.         this.age = age;
8.         this.name = name;
9.         this.address = address;
```

```
10.         }
11.     }
12.     class Employee extends Person
13.     {
14.         float salary;
15.         public Employee(int age, String name, String address, float salary)
16.         {
17.             super(age,name,address);
18.             this.salary = salary;
19.         }
20.     }
21.     public class Test
22.     {
23.         public static void main (String args[])
24.         {
25.             Employee e = new Employee(22, "Mukesh", "Delhi", 90000);
26.             System.out.println("Name: "+e.name+" Salary: "+e.salary+" Age: "+e.age+" Address: "+
    e.address);
27.         }
28.     }
```

Output

```
Name: Mukesh Salary: 90000.0 Age: 22 Address: Delhi
```

---

67) What are the main uses of the super keyword?

There are the following uses of super keyword.

- super can be used to refer to the immediate parent class instance variable.
- super can be used to invoke the immediate parent class method.
- super() can be used to invoke immediate parent class constructor.

---

68) What are the differences between this and super keyword?

There are the following differences between this and super keyword.

- The super keyword always points to the parent class contexts whereas this keyword always points to the current class context.
- The super keyword is primarily used for initializing the base class variables within the derived class constructor whereas this keyword primarily used to differentiate between local and instance variables when passed in the class constructor.

- The super and this must be the first statement inside constructor otherwise the compiler will throw an error.

---

69) What is the output of the following Java program?

```
1. class Person
2. {
3.     public Person()
4.     {
5.         System.out.println("Person class constructor called");
6.     }
7. }
8. public class Employee extends Person
9. {
10.         public Employee()
11.         {
12.             System.out.println("Employee class constructor called");
13.         }
14.         public static void main (String args[])
15.         {
```

```
16.             Employee e = new Employee();
17.         }
18.     }
```

Output

```
Person class constructor called
Employee class constructor called
```

Explanation

The super() is implicitly invoked by the compiler if no super() or this() is included explicitly within the derived class constructor. Therefore, in this case, The Person class constructor is called first and then the Employee class constructor is called.

---

70) Can you use this() and super() both in a constructor?

No, because this() and super() must be the first statement in the class constructor.

Example:

```
1. public class Test{
2.     Test()
```

```
3.     {
4.         super();
5.         this();
6.         System.out.println("Test class object is created");
7.     }
8.     public static void main(String []args){
9.     Test t = new Test();
10.          }
11.      }
```

Output:

```
Test.java:5: error: call to this must be first statement in constructor
```

---

71)What is object cloning?

The object cloning is used to create the exact copy of an object. The clone() method of the Object class is used to clone an object. The **java.lang.Cloneable** interface must be implemented by the class whose object clone we want to create. If we don't implement Cloneable interface, clone() method generates CloneNotSupportedException.

```
1. protected Object clone() throws CloneNotSupportedException
```

2.

More Details.

Core Java - OOPs Concepts: Method Overloading Interview Questions

---

72) What is method overloading?

Method overloading is the polymorphism technique which allows us to create multiple methods with the same name but different signature. We can achieve method overloading in two ways.

- Changing the number of arguments
- Changing the return type

Method overloading increases the readability of the program. Method overloading is performed to figure out the program quickly.

More Details.

---

73) Why is method overloading not possible by changing the return type in java?

In Java, method overloading is not possible by changing the return type of the program due to avoid the ambiguity.

```
1. class Adder{
2. static int add(int a,int b){return a+b;}
3. static double add(int a,int b){return a+b;}
4. }
5. class TestOverloading3{
6. public static void main(String[] args){
7. System.out.println(Adder.add(11,11));//ambiguity
8. }}
```

Test it Now

Output:

```
Compile Time Error: method add(int, int) is already defined in class Adder
```

More Details.

---

74) Can we overload the methods by making them static?

No, We cannot overload the methods by just applying the static keyword to them(number of parameters and types are the same). Consider the following example.

```
1. public class Animal
2. {
3.     void consume(int a)
4.     {
5.         System.out.println(a+" consumed!!");
6.     }
7.     static void consume(int a)
8.     {
9.         System.out.println("consumed static "+a);
10.         }
11.         public static void main (String args[])
12.         {
13.             Animal a = new Animal();
14.             a.consume(10);
15.             Animal.consume(20);
16.         }
17.     }
```

Output

```
Animal.java:7: error: method consume(int) is already defined in class Animal
    static void consume(int a)
```

```
            ^
Animal.java:15: error: non-static method consume(int) cannot be referenced from a static context
        Animal.consume(20);
              ^
2 errors
```

---

75) Can we overload the main() method?

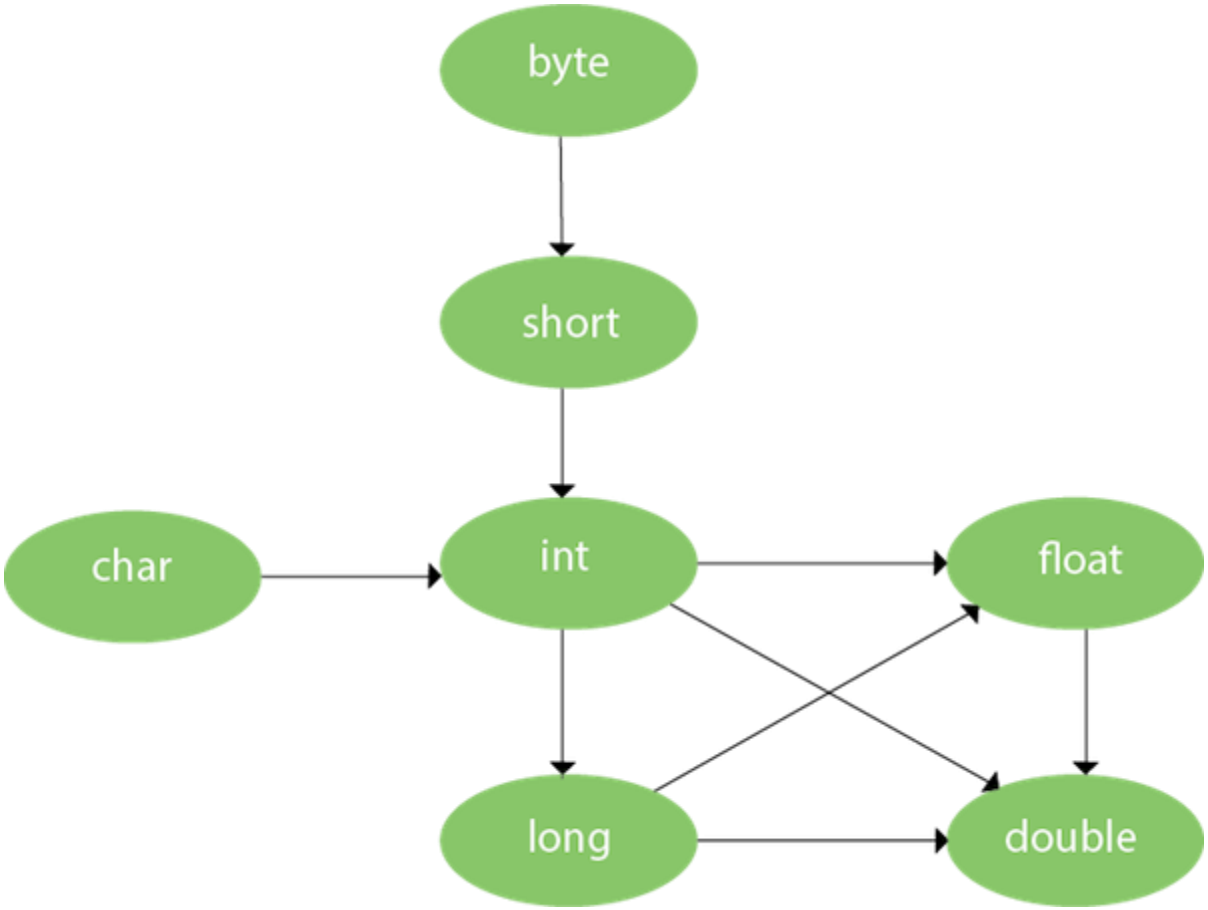
Yes, we can have any number of main methods in a Java program by using method overloading.

More Details.

---

76) What is method overloading with type promotion?

By Type promotion is method overloading, we mean that one data type can be promoted to another implicitly if no exact matching is found.



As displayed in the above diagram, the byte can be promoted to short, int, long, float or double. The short datatype can be promoted to int, long, float or double. The char datatype can be promoted to int, long, float or double and so on. Consider the following example.

```
1. class OverloadingCalculation1{
2.   void sum(int a,long b){System.out.println(a+b);}
3.   void sum(int a,int b,int c){System.out.println(a+b+c);}
```

```
4.
5.   public static void main(String args[]){
6.   OverloadingCalculation1 obj=new OverloadingCalculation1();
7.   obj.sum(20,20);//now second int literal will be promoted to long
8.   obj.sum(20,20,20);
9.   }
10.      }
```

Test it Now

Output

```
40
60
```

---

77) What is the output of the following Java program?

```
1. class OverloadingCalculation3{
2.   void sum(int a,long b){System.out.println("a method invoked");}
3.   void sum(long a,int b){System.out.println("b method invoked");}
4.
5.   public static void main(String args[]){
6.   OverloadingCalculation3 obj=new OverloadingCalculation3();
```

```
7.   obj.sum(20,20);//now ambiguity
8.   }
9. }
```

Output

```
OverloadingCalculation3.java:7: error: reference to sum is ambiguous
obj.sum(20,20);//now ambiguity
    ^
     both method sum(int,long) in OverloadingCalculation3
     and method sum(long,int) in OverloadingCalculation3 match
1 error
```

Explanation

There are two methods defined with the same name, i.e., sum. The first method accepts the integer and long type whereas the second method accepts long and the integer type. The parameter passed that are a = 20, b = 20. We can not tell that which method will be called as there is no clear differentiation mentioned between integer literal and long literal. This is the case of ambiguity. Therefore, the compiler will throw an error.

Core Java - OOPs Concepts: Method Overriding Interview Questions

78) What is method overriding:

If a subclass provides a specific implementation of a method that is already provided by its parent class, it is known as Method Overriding. It is used for runtime polymorphism and to implement the interface methods.

Rules for Method overriding

- The method must have the same name as in the parent class.
- The method must have the same signature as in the parent class.
- Two classes must have an IS-A relationship between them.

More Details.

---

79) Can we override the static method?

No, you can't override the static method because they are the part of the class, not the object.

---

80) Why can we not override static method?

It is because the static method is the part of the class, and it is bound with class whereas instance method is bound with the object, and static gets memory in class area, and instance gets memory in a heap.

---

81) Can we override the overloaded method?

Yes.

---

82) Difference between method Overloading and Overriding.

Method Overloading	Method Overriding
1) Method overloading increases the readability of the program.	Method overriding provides the specific implementation of the method that is already provided by its superclass.
2) Method overloading occurs within the class.	Method overriding occurs in two classes that have IS-A relationship between them.

3) In this case, the parameters must be different.	In this case, the parameters must be the same.

---

83) Can we override the private methods?

No, we cannot override the private methods because the scope of private methods is limited to the class and we cannot access them outside of the class.

---

84) Can we change the scope of the overridden method in the subclass?

Yes, we can change the scope of the overridden method in the subclass. However, we must notice that we cannot decrease the accessibility of the method. The following point must be taken care of while changing the accessibility of the method.

- The private can be changed to protected, public, or default.
- The protected can be changed to public or default.
- The default can be changed to public.
- The public will always remain public.

---

85) Can we modify the throws clause of the superclass method while overriding it in the subclass?

Yes, we can modify the throws clause of the superclass method while overriding it in the subclass. However, there are some rules which are to be followed while overriding in case of exception handling.

- If the superclass method does not declare an exception, subclass overridden method cannot declare the checked exception, but it can declare the unchecked exception.
- If the superclass method declares an exception, subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception.

---

86) What is the output of the following Java program?

```
1. class Base
2. {
3.     void method(int a)
4.     {
5.         System.out.println("Base class method called with integer a = "+a);
6.     }
7.
8.     void method(double d)
```

```
9.    {
10.           System.out.println("Base class method called with double d ="+d);
11.       }
12.    }
13.
14.    class Derived extends Base
15.    {
16.       @Override
17.       void method(double d)
18.       {
19.          System.out.println("Derived class method called with double d ="+d);
20.       }
21.    }
22.
23.    public class Main
24.    {
25.       public static void main(String[] args)
26.       {
27.          new Derived().method(10);
28.       }
29.    }
```

Output

```
Base class method called with integer a = 10
```

Explanation

The method() is overloaded in class Base whereas it is derived in class Derived with the double type as the parameter. In the method call, the integer is passed.

87) Can you have virtual functions in Java?

Yes, all functions in Java are virtual by default.

---

88) What is covariant return type?

Now, since java5, it is possible to override any method by changing the return type if the return type of the subclass overriding method is subclass type. It is known as covariant return type. The covariant return type specifies that the return type may vary in the same direction as the subclass.

```
1. class A{
2. A get(){return this;}
3. }
4.
```

```
5. class B1 extends A{
6. B1 get(){return this;}
7. void message(){System.out.println("welcome to covariant return type");}
8.
9. public static void main(String args[]){
10.     new B1().get().message();
11.     }
12.     }
```

Test it Now

Output: welcome to covariant return type

More Details.

---

89) What is the output of the following Java program?

```
1. class Base
2. {
3.     public void baseMethod()
4.     {
5.         System.out.println("BaseMethod called ...");
6.     }
```

```
7. }
8. class Derived extends Base
9. {
10.         public void baseMethod()
11.         {
12.             System.out.println("Derived method called ...");
13.         }
14.     }
15.     public class Test
16.     {
17.         public static void main (String args[])
18.         {
19.             Base b = new Derived();
20.             b.baseMethod();
21.         }
22.     }
```

Output

```
Derived method called ...
```

Explanation

The method of Base class, i.e., baseMethod() is overridden in Derived class. In Test class, the reference variable b (of type Base class) refers to the instance of the Derived class. Here, Runtime polymorphism is achieved between class Base and Derived. At compile time, the presence of method baseMethod checked in Base class, If it presence then the program compiled otherwise the compiler error will be shown. In this case, baseMethod is present in Base class; therefore, it is compiled successfully. However, at runtime, It checks whether the baseMethod has been overridden by Derived class, if so then the Derived class method is called otherwise Base class method is called. In this case, the Derived class overrides the baseMethod; therefore, the Derived class method is called.

Core Java - OOPs Concepts: final keyword Interview Questions

---

90) What is the final variable?

In Java, the final variable is used to restrict the user from updating it. If we initialize the final variable, we can't change its value. In other words, we can say that the final variable once assigned to a value, can never be changed after that. The final variable which is not assigned to any value can only be assigned through the class constructor.


```
1. class Bike9{
2.  final int speedlimit=90;//final variable
3.  void run(){
4.   speedlimit=400;
5.  }
6.  public static void main(String args[]){
7.  Bike9 obj=new  Bike9();
8.  obj.run();
9.  }
10.       }//end of class
```

Test it Now

Output:Compile Time Error

More Details.

---

91) What is the final method?

If we change any method to a final method, we can't override it. More Details.

```
1. class Bike{
2.   final void run(){System.out.println("running");}
3. }
4.
5. class Honda extends Bike{
6.    void run(){System.out.println("running safely with 100kmph");}
7.
8.    public static void main(String args[]){
9.    Honda honda= new Honda();
10.         honda.run();
11.         }
12.      }
```

Test it Now

Output:Compile Time Error

---

92) What is the final class?

If we make any class final, we can't inherit it into any of the subclasses.

```
1. final class Bike{}
2.
3. class Honda1 extends Bike{
4.   void run(){System.out.println("running safely with 100kmph");}
5.
6.   public static void main(String args[]){
7.   Honda1 honda= new Honda1();
8.   honda.run();
9.   }
10.      }
```

Test it Now

Output:Compile Time Error

More Details.

---

93) What is the final blank variable?

A final variable, not initialized at the time of declaration, is known as the final blank variable. We can't initialize the final blank variable directly. Instead, we have to initialize it by using the class constructor. It is useful in the case when the user has some data which must not be changed by others, for example, PAN Number. Consider the following example:

```
1. class Student{
2. int id;
3. String name;
4. final String PAN_CARD_NUMBER;
5. ...
6. }
```

More Details.

---

94) Can we initialize the final blank variable?

Yes, if it is not static, we can initialize it in the constructor. If it is static blank final variable, it can be initialized only in the static block. More Details.

---

95) Can you declare the main method as final?

Yes, We can declare the main method as public static final void main(String[] args){}.

---

96) What is the output of the following Java program?

```
1. class Main {
2.  public static void main(String args[]){
3.    final int i;
4.    i = 20;
5.    System.out.println(i);
6.  }
7. }
```

Output

```
20
```

Explanation

Since i is the blank final variable. It can be initialized only once. We have initialized it to 20. Therefore, 20 will be printed.

---

97) What is the output of the following Java program?

```
1. class Base
2. {
3.     protected final void getInfo()
4.     {
5.         System.out.println("method of Base class");
6.     }
7. }
8.
9. public class Derived extends Base
10.     {
11.         protected final void getInfo()
12.         {
13.             System.out.println("method of Derived class");
14.         }
15.         public static void main(String[] args)
16.         {
17.             Base obj = new Base();
18.             obj.getInfo();
19.         }
```

```
20.     }
```

Output

```
        Derived.java:11: error: getInfo() in Derived cannot override getInfo() in Base
    protected final void getInfo()
                         ^
  overridden method is final
1 error
```

Explanation

The getDetails() method is final; therefore it can not be overridden in the subclass.

---

98) Can we declare a constructor as final?

The constructor can never be declared as final because it is never inherited. Constructors are not ordinary methods; therefore, there is no sense to declare constructors as final. However, if you try to do so, The compiler will throw an error.

---

99) Can we declare an interface as final?

No, we cannot declare an interface as final because the interface must be implemented by some class to provide its definition. Therefore, there is no sense to make an interface final. However, if you try to do so, the compiler will show an error.

OPERATING SYSTEM

1) Explain the main purpose of an operating system?

Operating systems exist for two main purposes. One is that it is designed to make sure a computer system performs well by managing its computational activities. Another is that it provides an environment for the development and execution of programs.

2) What is demand paging?

Demand paging is referred when not all of a process's pages are in the RAM, then the OS brings the missing(and required) pages from the disk into the RAM.

3) What are the advantages of a multiprocessor system?

With an increased number of processors, there is a considerable increase in throughput. It can also save more money because they can share resources. Finally, overall reliability is increased as well.

4) What is kernel?

A kernel is the core of every operating system. It connects applications to the actual processing of data. It also manages all communications between software and hardware components to ensure usability and reliability.

5) What are real-time systems?

Real-time systems are used when rigid time requirements have been placed on the operation of a processor. It has well defined and fixed time constraints.

6) What is a virtual memory?

Virtual memory is a memory management technique for letting processes execute outside of memory. This is very useful especially is an executing program cannot fit in the physical memory.

7) Describe the objective of multiprogramming.

The main objective of multiprogramming is to have a process running at all times. With this design, CPU utilization is said to be maximized.

8) What is time- sharing system?

In a Time-sharing system, the CPU executes multiple jobs by switching among them, also known as multitasking. This process happens so fast that users can interact with each program while it is running.

9) What is SMP?

SMP is a short form of Symmetric Multi-Processing. It is the most common type of multiple-processor systems. In this system, each processor runs an identical copy of the operating system, and these copies communicate with one another as needed.

10) How are server systems classified?

Server systems can be classified as either computer-server systems or file server systems. In the first case, an interface is made available for clients to send requests to perform an action. In the second case, provisions are available for clients to create, access and update files.

11) What is asymmetric clustering?

In asymmetric clustering, a machine is in a state known as hot standby mode where it does nothing but to monitor the active server. That machine takes the active server’s role should the server fails.

12) What is a thread?

A thread is a basic unit of CPU utilization. In general, a thread is composed of a thread ID, program counter, register set, and the stack.

13) Give some benefits of multithreaded programming.

– there is increased responsiveness to the user
– resource sharing within the process
– economy
– utilization of multiprocessing architecture

14) Briefly explain FCFS.

FCFS stands for First-come, first-served. It is one type of scheduling algorithm. In this scheme, the process that requests the CPU first is allocated the CPU first. Implementation is managed by a FIFO queue.

15) What is RR scheduling algorithm?

RR (round-robin) scheduling algorithm is primarily aimed for time-sharing systems. A circular queue is a setup in such a way that the CPU scheduler goes around that queue, allocating CPU to each process for a time interval of up to around 10 to 100 milliseconds.

16) What are necessary conditions which can lead to a deadlock situation in a system?

Deadlock situations occur when four conditions occur simultaneously in a system: Mutual exclusion; Hold and Wait; No pre-emption; and Circular wait.

17) Enumerate the different RAID levels.

RAID 0 – Non-redundant striping

RAID 1 – Mirrored Disks

RAID 2 – Memory-style error-correcting codes

RAID 3 – Bit-interleaved Parity

RAID 4 – Block-interleaved Parity

RAID 5 – Block-interleaved distributed Parity

RAID 6 – P+Q Redundancy

18) Describe Banker's algorithm

Banker's algorithm is one form of deadlock-avoidance in a system. It gets its name from a banking system wherein the bank never allocates available cash in such a way that it can no longer satisfy the needs of all of its customers.

19) What factors determine whether a detection-algorithm must be utilized in a deadlock avoidance system?

One is that it depends on how often a deadlock is likely to occur under the implementation of this algorithm. The other has to do with how many processes will be affected by deadlock when this algorithm is applied.

20) State the main difference between logical from physical address space.

Logical address refers to the address that is generated by the CPU. On the other hand, physical address refers to the address that is seen by the memory unit.

21) How does dynamic loading aid in better memory space utilization?

With dynamic loading, a routine is not loaded until it is called. This method is especially useful when large amounts of code are needed in order to handle infrequently occurring cases such as error routines.

22) What are overlays?

Overlays are used to enable a process to be larger than the amount of memory allocated to it. The basic idea of this is that only instructions and data that are needed at any given time are kept in memory.

23) What is the basic function of paging?

Paging is a memory management scheme that permits the physical address space of a process to be non-contiguous. It avoids the considerable problem of having to fit varied sized memory chunks onto the backing store.

24) What is fragmentation?

Fragmentation is memory wasted. It can be internal if we are dealing with systems that have fixed-sized allocation units, or external if we are dealing with systems that have variable-sized allocation units.

25) How does swapping result in better memory management?

During regular intervals that are set by the operating system, processes can be copied from main memory to a backing store, and then copied back later. Swapping allows more operations to be run that can fit into memory at one time.

26) Give an example of a Process State.

– New State – means a process is being created
– Running – means instructions are being executed
– Waiting – means a process is waiting for certain conditions or events to occur
– Ready – means a process is waiting for an instruction from the main processor
– Terminate – means a process is stopped abruptly

27) What is a socket?

A socket provides a connection between two applications. Each endpoint of a communication is a socket.

28) What is Direct Access Method?

Direct Access method is based on a disk model of a file, such that it is viewed as a numbered sequence of blocks or records. It allows arbitrary blocks to be read or written. Direct access is advantageous when accessing large amounts of information.

29) When does thrashing occur?

Thrashing refers to an instance of high paging activity. This happens when it is spending more time paging instead of executing.

30) What is the best page size when designing an operating system?

The best paging size varies from system to system, so there is no single best when it comes to page size. There are different factors to consider in order to come up with a suitable page size, such as page table, paging time, and its effect on the overall efficiency of the operating system.

31) When designing the file structure for an operating system, what attributes are considered?

Typically, the different attributes for a file structure are naming, identifier, supported file types, and location for the files, size, and level of protection.

32) What is root partition?

Root partition is where the operating system kernel is located. It also contains other potentially important system files that are mounted during boot time.

33) What are device drivers?

Device drivers provide a standard means of representing I/O devices that maybe manufactured by different companies. This prevents conflicts whenever such devices are incorporated in a systems unit.

34) What are the primary functions of VFS?

VFS, or Virtual File System, separate file system generic operations from their implementation by defining a clean VFS interface. It is based on a file-representation structure known as vnode, which contains a numerical designator needed to support network file systems.

35) What are the different types of CPU registers in a typical operating system design?

– Accumulators
– Index Registers
– Stack Pointer
– General Purpose Registers

36) What is the purpose of an I/O status information?

I/O status information provides information about which I/O devices are to be allocated for a particular process. It also shows which files are opened, and other I/O device state.

37) What is multitasking?

Multitasking is the process within an operating system that allows the user to run several applications at the same time. However, only one application is active at a time for user interaction, although some applications can run "behind the scene".

38) Explain pros and cons of a command line interface?

A command line interface allows the user to type in commands that can immediately provide results. Many seasoned computer users are well accustomed to using the command line because they find it quicker and simpler.

However, the main problem with a command line interface is that users have to be familiar with the commands, including the switches and parameters that come with it. This is a downside for people who are not fond of memorizing commands.

39) What is caching?

Caching is the processing of utilizing a region of fast memory for a limited data and process. A cache memory is usually much efficient because of its high access speed.

40) What is spooling?

Spooling is normally associated with printing. When different applications want to send an output to the printer at the same time, spooling takes all of these print jobs into a disk file and queues them accordingly to the printer.

41) What is an Assembler?

An assembler acts as a translator for low-level language. Assembly codes written using mnemonic commands are translated by the Assembler into machine language.

42) What are interrupts?

Interrupts are part of a hardware mechanism that sends a notification to the CPU when it wants to gain access to a particular resource. An interrupt handler receives this interrupt signal and "tells" the processor to take action based on the interrupt request.

43) What is GUI?

GUI is short for Graphical User Interface. It provides users with an interface wherein actions can be performed by interacting with icons and graphical symbols. People find it easier to interact with the computer when in a GUI especially when using the mouse. Instead of having to remember and type commands, users click on buttons to perform a process.

44) What is pre-emptive multitasking?

Pre-emptive multitasking allows an operating system to switch between software programs. This, in turn, allows multiple programs to run without necessarily taking complete control over the processor and resulting in system crashes.

45) Why partitioning and formatting is a prerequisite to installing an operating system?

Partitioning and formatting create a preparatory environment on the drive so that the operating system can be copied and installed properly. This includes allocating space on the drive, designating a drive name, determining and creating the appropriate file system and structure.

46) What is plumbing/piping?

It is the process of using the output of one program as an input to another. For example, instead of sending the listing of a folder or drive to the main screen, it can be piped and sent to a file, or sent to the printer to produce a hard copy.

47) What is NOS?

NOS is short for Network Operating System. It is a specialized software that will allow a computer to communicate with other devices over the network, including file/folder sharing.

48) Differentiate internal commands from external commands.

Internal commands are built-in commands that are already part of the operating system. External commands are separate file programs that are stored in a separate folder or directory.

49) Under DOS, what command will you type when you want to list down the files in a directory, and at the same time pause after every screen output?

a) dir /w

b) dir /p

c) dir /s

d) dir /w /p

Answer: d) dir /w /p

50) How would a file name EXAMPLEFILE.TXT appear when viewed under the DOS command console operating in Windows 98?

The filename would appear as EXAMPL~1.TXT. The reason behind this is that filenames under this operating system are limited to 8 characters when working under DOS environment.

51) What are the different scheduling algorithms?

First-Come, First-Served (FCFS) Scheduling.
Shortest-Job-Next (SJN) Scheduling.
Priority Scheduling.
Shortest Remaining Time.
Round Robin(RR) Scheduling.
Multiple-Level Queues Scheduling.

52) What is deadlock?

Deadlock is a situation when two or more processes wait for each other to finish and none of them ever finish. Consider an example when two trains are coming toward each other on same track and there is only one track, none of the trains can move once they are in front of each other. Similar situation occurs in operating systems when there are two or more processes hold some resources and wait for resources held by other(s).

53) What are the necessary conditions for deadlock?

Mutual Exclusion: There is a resource that cannot be shared.
Hold and Wait: A process is holding at least one resource and waiting for another resource which is with some other process.
No Preemption: The operating system is not allowed to take a resource back from a process until process gives it back.
Circular Wait: A set of processes are waiting for each other in circular form.

54) What Is Kernel?

Kernel is the core of every operating system. It connects applications to the actual processing of data. It also manages all communications between software and hardware components to ensure usability and reliability.

55)How Are Server Systems Classified?

Server systems can be classified as either computer-server systems or file server systems. In the first case, an interface is made available for clients to send requests to perform an action. In the second case, provisions are available for clients to create, access and update files.

56) Differentiate Logical from Physical Address Space.?

Logical address refers to the address that is generated by the CPU. On the other hand, physical address refers to the address that is seen by the memory unit.

57) Question 20. How Does Dynamic Loading Aid In Better Memory Space Utilization?

With dynamic loading, a routine is not loaded until it is called. This method is especially useful when large amounts of code are needed in order to handle infrequently occurring cases such as error routines.

58) What Are Overlays?

Overlays are used to enable a process to be larger than the amount of memory allocated to it. The basic idea of this is that only instructions and data that are needed at any given time are kept in memory.

59) How Does Swapping Result in Better Memory Management?

During regular intervals that are set by the operating system, processes can be copied from main memory to a backing store, and then copied back later. Swapping allows more processes to be run that can fit into memory at one time.

60)What Is the Best Page Size When Designing An Operating System?

The best paging size varies from system to system, so there is no single best when it comes to page size. There are different factors to consider in order to come up with a suitable page size, such as page table, paging time, and its effect on the overall efficiency of the operating system.

61)When Designing the File Structure for An Operating System, What Attributes Are Considered?

Typically, the different attributes for a file structure are naming, identifier, supported file types, and location for the files, size, and level of protection.

62)What Are Device Drivers?

Device drivers provides a standard means of representing I/O devices that maybe manufactured by different companies. This prevents conflicts whenever such devices are incorporated in a systems unit.

63) What Are the Different Types of CPU Registers In A Typical Operating System Design?

- Accumulators
- Index Registers

- Stack Pointer
- General Purpose Registers

64)What Is the Purpose ofan I/o Status Information?

I/O status information provides info about which I/O devices are to be allocated for a particular process. It also shows which files are opened, and other I/O device state.

65) What Are Some Pros and Cons ofa Command Line Interface?

A command line interface allows the user to type in commands that can immediately provide results. Many seasoned computer users are well accustomed to using the command line because they find it quicker and simpler. The main problem with a command line interface is that users have to be familiar with the commands, including the switches and parameters that come with it. This is a downside for people who are not fond of memorizing commands.

66)What Is an Assembler?

An assembler acts as a translator for low level language. Assembly codes, written using mnemonic commands are translated by the Assembler into machine language.

67) What Are Interrupts?

Interrupts are part of a hardware mechanism that sends a notification to the CPU when it wants to gain access to a particular resource. An interrupt handler receives this interrupt signal and "tells" the processor to take action based on the interrupt request.

68) What Is GUI?

GUI is short for Graphical User Interface. It provides users with an interface wherein actions can be performed by interacting with icons and graphical symbols. People find it easier to interact with the computer when in a GUI especially when using the mouse. Instead of having to remember and type commands, users just click on buttons to perform a process.

69) Differentiate Internal Commands from External Commands.

Internal commands are built-in commands that are already part of the operating system.
External commands are separate file programs that are stored in a separate folder or directory.

70) Describe system calls and its type

System calls works as a mediator between user program and service provided by operating system. In actual situation, functions that make up an API (application program interface) typically invoke the actual system calls on behalf of the application programmer.

1	Process control	Create process, terminate process, end, allocate and free memory etc
2	File manipulation	Create file, delete file, open file, close file, read, write.
3	Device manipulation	request device, release device, read, write, reposition, get device attributes, set device attributes etc.
4	Information maintenance	get or set process, file, or device attributes
5	Communications	Send, receive messages, transfer status information

71) Explain Booting the system and Bootstrap program in operating system.

The procedure of starting a computer by loading the kernel is known as booting the system. When a user first turns on or booted the computer, it needs some initial program to run. This initial program is known as Bootstrap Program. It is stored in read-only memory (ROM) or electrically erasable programmable read-only memory (EEPROM). Bootstrap program locates the kernel and loads it into main memory and starts its execution.

72) Describe Main memory and Secondary memory storage in brief.

Main memory is also called random access memory (RAM). CPU can access Main memory directly. Data access from main memory is much faster than Secondary memory. It is implemented in a semiconductor technology, called dynamic random-access memory (DRAM).

Main memory is usually too small to store all needed programs. It is a volatile storage device that loses its contents when power is turned off. Secondary memory can store large amount of data and programs permanently. Magnetic disk is the most common secondary storage device. If a user wants to execute any program it should come from secondary memory to main memory because CPU can access main memory directly.

73) What are the different types of Kernel?

Kernels are basically of two types:

a. Monolithic Kernels - In this architecture of kernel, all the system services were packaged into a single system module which lead to poor maintainability and huge size of kernel.

b. Microkernels - They follow the modular approach of architecture. Maintainability became easier with this model as only the concerned module is to be altered and loaded for every function. This model also keeps a tab on the ever-growing code size of the kernel.

74) What are the disadvantages of Microkernels?

Following are the main disadvantages of Microkernels. Usually these disadvantages are situation based.

a. Larger running memory footprint

b. Performance loss due to the requirement of more software for interfacing.

c. Difficulty in fixing the messaging bugs.

d. Complicated process management.

75) What is a command interpreter?

It is a program that interprets the command input through keyboard or command batch file. It helps the user to interact with the OS and trigger the required system programs or execute some user application.

Command interpreter is also referred to as:

Control card interpreter

Command line interpreter

Console command processor

Shell

76) What do you know about interrupt?

- Interrupt can be understood as a signal from a device causing context switch.

- To handle the interrupts, interrupt handlers or service routines are required.

- The address of each Interrupt service routine is provided in a list which is maintained in interrupt vector.

77) What is a daemon?

- Daemon - Disk and execution monitor, is a process that runs in the background without user's interaction. They usually start at the booting time and terminate when the system is shut down.

78) How would you identify daemons in Unix?

- The name of daemons usually end with 'd' at the end in Unix.

- For e.g. httpd, named, lpd.

79) What do you mean by a zombie process?

- These are dead processes which are not yet removed from the process table.

- It happens when the parent process has terminated while the child process is still running. This child process now stays as a zombie.

80) What do you know about a Pipe? When is it used?

- It is an IPC mechanism used for one way communication between two processes which are related.

- A single process doesn't need to use pipe. It is used when two process wish to communicate one-way.

81) What is a named pipe?

- A traditional pipe is unnamed and can be used only for the communication of related process. If unrelated processes are required to communicate - named pipes are required.

- It is a pipe whose access point is a file available on the file system. When this file is opened for reading, a process is granted access to the reading end of the pipe. Similarly, when the file is opened for writing, the process is granted access to writing end of the pipe.

- A named pipe is also referred to as FIFO or named FIFO.

82) What are the various IPC mechanisms?

IPC - Inter Process Communication.

Various IPC mechanisms are:

a. Sockets

b. Pipes

c. Shared memory

d. Signals

e. Message Queues

83) What is a semaphore?

- A semaphore is a hardware or a software tag variable whose value indicates the status of a common resource.

- Its purpose is to lock the common resource being used. A process which needs the resource will check the semaphore to determine the status of the resource followed by the decision for proceeding.

- In multitasking operating systems, the activities are synchronized by using the semaphore techniques.

84) What kind of operations are possible on a semaphore?

Two kind of operations are possible on a semaphore - 'wait' and 'signal'.

85) What is context switching?

- Context is associated with each process encompassing all the information describing the current execution state of the process

- When the OS saves the context of program that is currently running and restores the context of the next ready to run process, it is called as context switching.

- It is important for multitasking OS.

86) Tell us something about Mutex.

- Mutex - ‘Mutual Exclusion Lock’ is a lock which protects access to shared data resource.

- Threads can create and initialize a mutex to be used later.

- Before entering a critical region the mutex is locked. It is unlocked after exiting the critical region. If any thread tries to lock the mutex during this time, it can't do so.

87) What is a critical section?

It is a section of code which can be executed only by one process at a time.

88) What is synchronization? What are the different synchronization mechanisms?

Synchronization means controlling access to a resource that is available to two or more threads or process. Different synchronization mechanisms are:

- Mutex

- Semaphores

- Monitors

- Condition variables

- Critical regions

- Read/ Write locks

89) What is the basic difference between pre-emptive and non-pre-emptive scheduling?

Pre-emptive scheduling allows interruption of a process while it is executing and taking the CPU to another process while non-pre-emptive scheduling ensures that a process keeps the CPU under control until it has completed execution.

90) Is non-pre-emptive scheduling frequently used in a computer? Why?

No, it is rarely used for the reasons mentioned below:

- It can not ensure that each user gets a share of CPU regularly.

- The idle time with this increases reducing the efficiency and overall performance of the system.

- It allows program to run indefinitely which means that other processes have to wait for very long.

91) Explain condition variable.

- These are synchronization objects which help threads wait for particular conditions to occur.

- Without condition variable, the thread has to continuously check the condition which is very costly on the resources.

- Condition variable allows the thread to sleep and wait for the condition variable to give it a signal.

92) What are read-write locks?

- Read - write locks provide simultaneous read access to many threads while the write access stays with one thread at a time. They are especially useful in protecting the data that is not frequently written but read simultaneously by many threads.

- They are slower than mutexes.

93) What is a deadlock?

- It is a condition where a group of two or more waiting for the resources currently in use by other processes of the same group.

- In this situation every process is waiting for an event to be triggered by another process of the group.

- Since no thread can free up the resource a deadlock occurs and the application hangs.

94) What are the necessary conditions for deadlock to occur?

a. At least one resource should be occupied in a non-sharable condition.

b. A process holding at least one resource is waiting for more resources currently in use by other processes.

c. It is not possible to pre-empt the resource.

d. There exists a circular wait for processes.

95) Name the functions constituting the OS's memory management.

- Memory allocation and de-allocation

- Integrity maintenance

- Swapping

- Virtual memory

96) Name the different types of memory?

a. Main memory also called primary memory or RAM

b. Secondary memory or backing storage

c. Cache

d. Internal process memory

97) Throw some light on Internal Process Memory.

- This memory consists of a set of high-speed registers. They work as temporary storage for instructions and data.

98) Explain compaction.

During the process of loading and removal of process into and out of the memory, the free memory gets broken into smaller pieces. These pieces lie scattered in the memory. Compaction means movement of these pieces close to each other to form a larger chunk of memory which works as a resource to run larger processes.

99) What are page frames?

Page frames are the fixed size contiguous areas into which the main memory is divided by the virtual memory.

100) What are pages?

- Pages are same sized pieces of logical memory of a program. Usually they range from 4 KB to 8 KB depending on the addressing hardware of the machine.

- Pages improve the overall system performance and reduces requirement of physical storage as the data is read in 'page' units.

DataBase Management System

Q1) Define Database.

A prearranged collection of figures known as data is called database.

Q2) What is DBMS?

Database Management Systems (DBMS) are applications designed especially which enable user interaction with other applications.

Q3) What are the various kinds of interactions catered by DBMS?

The various kind of interactions catered by DBMS are:

- Data definition
- Update
- Retrieval
- Administration

Q4) Segregate database technology's development.

The development of database technology is divided into:

- Structure or data model
- Navigational model
- SQL/ relational model

Q5) Who proposed the relational model?

Edgar F. Codd proposed the relational model in 1970.

Q6) What are the features of Database language?

A database language may also incorporate features like: DBMS-specific Configuration and management of storage engine Computations to modification of query results by computations, like summing, counting, averaging, grouping, sorting and cross-referencing Constraint enforcement Application Programming Interface

Q7) What do database languages do?

As special-purpose languages, they have:

- Data definition language
- Data manipulation language
- Query language

Q8) Define database model.

A data model determining fundamentally how data can be stored, manipulated and organised and the structure of the database logically is called database model.

Q9) What is SQL?

Structured Query Language (SQL) being ANSI standard language updates database and commands for accessing.

Q10) Enlist the various relationships of database.

The various relationships of database are:

- One-to-one: Single table having drawn relationship with another table having similar kind of columns.
- One-to-many: Two tables having primary and foreign key relation.
- Many-to-many: Junction table having many tables related to many tables.

Q11) Define Normalization.

Organized data void of inconsistent dependency and redundancy within a database is called normalization.

Q12) Enlist the advantages of normalizing database.

Advantages of normalizing database are:

- No duplicate entries
- Saves storage space
- Boasts the query performances.

Q13) Define Denormalization.

Boosting up database performance, adding of redundant data which in turn helps rid of complex data is called denormalization.

Q14) Define DDL and DML.

Managing properties and attributes of database is called Data Definition Language(DDL).

Manipulating data in a database such as inserting, updating, deleting is defined as Data Manipulation Language. (DML)

Q15) Enlist some commands of DDL.

They are:

CREATE:

Create is used in the CREATE TABLE statement. Syntax is:

```
CREATE TABLE [column name] ( [column definitions] ) [ table parameters]
```

ALTER:

It helps in modification of an existing object of database. Its syntax is:

```
ALTER objecttype objectname parameters.
```

DROP:

It destroys an existing database, index, table or view. Its syntax is:

```
DROP objecttype objectname.
```

Q16) Define Union All operator and Union.

Full recordings of two tables is Union All operator.A distinct recording of two tables is Union.

Q17) Define cursor.

A database object which helps in manipulating data row by row representing a result set is called cursor.

Q18) Enlist the cursor types.

They are:

- Dynamic: it reflects changes while scrolling.
- Static: doesn't reflect changes while scrolling and works on recording of snapshot.
- Keyset: data modification without reflection of new data is seen.

Q19) Enlist the types of cursor.

They types of cursor are:

- Implicit cursor: Declared automatically as soon as the execution of SQL takes place without the awareness of the user.
- Explicit cursor: Defined by PL/ SQL which handles query in more than one row.

Q20) Define sub-query.

A query contained by a query is called Sub-query.

Q21) Why is group-clause used?

Group-clause uses aggregate values to be derived by collecting similar data.

Q22) Compare Non-clustered and clustered index

Both having B-tree structure, non-clustered index has data pointers enabling one table many non-clustered indexes while clustered index is distinct for every table.

Q23) Define Aggregate functions.

Functions which operate against a collection of values and returning single value is called aggregate functions

Q24) Define Scalar functions.

Scalar function is depended on the argument given and returns sole value.

Q25) What restrictions can you apply when you are creating views?

Restrictions that are applied are:

- Only the current database can have views.
- You are not liable to change any computed value in any particular view.
- Integrity constants decide the functionality of INSERT and DELETE.
- Full-text index definitions cannot be applied.
- Temporary views cannot be created.

- Temporary tables cannot contain views.
- No association with DEFAULT definitions.
- Triggers such as INSTEAD OF is associated with views.

Q26) Define "correlated subqueries".

A 'correlated subquery' is a sort of sub query but correlated subquery is reliant on another query for a value that is returned. In case of execution, the sub query is executed first and then the correlated query.

Q27) Define Data Warehousing.

Storage and access of data from the central location in order to take some strategic decision is called Data Warehousing. Enterprise management is used for managing the information whose framework is known as Data Warehousing.

Q28) Define Join and enlist its types.

Joins help in explaining the relation between different tables. They also enable you to select data with relation to data in another table.

The various types are:

- INNER JOINs: Blank rows are left in the middle while more than equal to two tables are joined.
- OUTER JOINs: Divided into Left Outer Join and Right Outer Join. Blank rows are left at the specified side by joining tables in other side.

Other joins are CROSS JOINs, NATURAL JOINs, EQUI JOIN and NON-EQUI JOIN.

Q29) What do you mean by Index hunting?

Indexes help in improving the speed as well as the query performance of database. The procedure of boosting the collection of indexes is named as Index hunting.

Q30) How does Index hunting help in improving query performance?

Index hunting helps in improving the speed as well as the query performance of database. The followed measures are achieved to do that:

- The query optimizer is used to coordinate the study of queries with the workload and the best use of queries suggested based on this.
- Index, query distribution along with their performance is observed to check the effect.
- Tuning databases to a small collection of problem queries is also recommended.

Q31) Enlist the disadvantages of query.

The disadvantages of query are:

- No indexes
- Stored procedures are excessively compiled.
- Triggers and procedures are without SET NOCOUNT ON.
- Complicated joins making up inadequately written query.
- Cursors and temporary tables showcase a bad presentation.

Q32) Enlist ways to efficiently code transactions.

Ways to efficiently code transactions:

- User input should not be allowed while transactions.
- While browsing, transactions must not be opened of data.
- Transactions must be kept as small as possible.
- Lower transaction segregation levels.
- Least information of data must be accessed while transacting.

Q33) What is Executive Plan?

Executive plan can be defined as:

- SQL Server caches collected procedure or the plan of query execution and used thereafter by subsequent calls.
- An important feature in relation to performance enhancement.
- Data execution plan can be viewed textually or graphically.

Q34) Define B-trees.

A data structure in the form of tree which stores sorted data and searches, insertions, sequential access and deletions are allowed in logarithmic time.

Q35) Differentiate Table Scan from Index Scan.

Iterating over all the table rows is called Table Scan while iterating over all the index items is defined as Index Scan.

Q36) What do you mean by Fill Factor concept with respect to indexes?

Fill Factor can be defined as being that value which defines the percentage of left space on every leaf-level page that is to be packed with data. 100 is the default value of Fill Factor.

Q37) Define Fragmentation.

Fragmentation can be defined as a database feature of server that promotes control on data which is stored at table level by the user.

Q38) Differentiate Nested Loop, Hash Join and Merge Join.

Nested loop (loop over loop)

An outer loop within an inner loop is formed consisting of fewer entries and then for individual entry, inner loop is individually processed.

E.g.

- Select col1.*, col2.* from coll, col2 where coll.col1=col2.col2;

It's processing takes place in this way:

For i in (select * from col1) loop

For j in (select * from col2 where col2=i.col1) loop

Results are displayed;

End of the loop;

End of the loop;

The Steps of nested loop are:

- Identify outer (driving) table
- Assign inner (driven) table to outer table.
- For every row of outer table, access the rows of inner table.

Nested Loops is executed from the inner to the outer as:

- outer_loop
- inner_loop
- Hash join

While joining large tables, the use of Hash Join is preferred.

Algorithm of Hash Join is divided into:

- Build: It is a hash table having in-memory which is present on the smaller table.
- Probe: this hash value of the hash table is applicable for each second row element.
- Sort merge join

Two independent sources of data are joined in sort merge join. They performance is better as compared to nested loop when the data volume is big enough but it is not good as hash joins generally.The full operation can be divided into parts of two:

Sort join operation :

Get first row R1 from input1

Get first row R2 from input2.

Merge join operation:

'while' is not present at either loop's end.

if R1 joins with R2

next row is got R2 from the input 2

return (R1, R2)

else if R1 < style=""> next row is got from R1 from input 1

else

next row is got from R2 from input 2

end of the loop

Q39) What is Database partitioning?

Division of logical database into independent complete units for improving its management, availability and performance is called Database partitioning.

Q40) Explain the importance of partitioning.

Splitting of one table which is large into smaller database entities logically is called database partitioning. Its benefits are:

- To improve query performance in situations dramatically when mostly rows which are heavily accessed are in one partition.
- Accessing large parts of a single partition
- Slower and cheaper storage media can be used for data which is seldom used.

Q41) Define Database system.

DBMS along with database is called Database system.

Q42) What do you mean by Query Evaluation Engine?

Query Evaluation Engine executes the low-level instructions that are generated by the compiler.

Q43) Define DDL Interpreter.

DDL statements are interpreted and recorded in tables called metadata.

Q44) Define Atomicity and Aggregation.

Atomicity: It's an all or none concept which enables the user to be assured of incomplete transactions to be taken care of. The actions involving incomplete transactions are left undone in DBMS.

Aggregation: The collected entities and their relationship are aggregated in this model. It is mainly used in expressing relationships within relationships.

Q45) Enlist the various transaction phases.

The various transaction phases are:

- Analysis Phase.
- Redo Phase
- Undo Phase

Q46) Define Object-oriented model.

Compilations of objects make up this model in which values are stored within instance variables which is inside the object. The object itself comprises bodies of object for its operation which are called methods. Objects containing same kind of variables and methods are called classes.

Q47) Define Entity.

It can be defined as being a 'thing' with an independent existence in the real world.

Q48) What do you mean by Entity type?

A set of entries having similar attributes are entity types.

Q49) Define Entity Set.

Compilation of all entries of any particular type of entry in the database is called Entity Set.

Q50) What do you mean by Entity type extension?

Compilation of similar entity types into one particular type which is grouped together as an entity set.

Q51) What is Oracle and what are its different editions?

Oracle is one of the popular database provided by Oracle Corporation, which works on relational management concepts and hence it is referred as Oracle RDBMS as well.
It is widely used for online transaction processing, data warehousing, and enterprise grid computing.

Q52) How will you identify Oracle Database Software Release?

Oracle follows a number of format for every release.

For Example, release 10.1.0.1.1 can be referred as below mentioned:

10:	Major	DB	Release	Number
1:	DB	Maintenance	Release	Number
0:	Application	Server	Release	Number
1:	Component	Specific	Release	Number

1: Platform Specific Release Number

Q53) How will you differentiate between VARCHAR & VARCHAR2?

A Both VARCHAR & VARCHAR2 are Oracle data types that are used to store character strings of variable length.

VARCHAR can store characters up to 2000 bytes while VARCHAR2 can store up to 4000 bytes.

VARCHAR will hold the space for characters defined during declaration even if all of them are not used whereas VARCHAR2 will release the unused space.

Q54) What is the difference between TRUNCATE & DELETE command?

QBoth the commands are used to remove data from a database.

The finer differences between the two include:

- TRUNCATE is a DDL operation while DELETE is a DML operation.
- TRUNCATE drops the structure of a database and hence cannot be rolled back while DELETE command can be rolled back.
- The TRUNCATE command will free the object storage space while the DELETE command does not.

Q55) What is meant by RAW datatype?

RAW datatype is used to store variable-length binary data or byte strings.

The difference between RAW & VARCHAR2 datatype is that PL/SQL does not recognize this data type and hence, cannot do any conversions when RAW data is transferred to different systems. This data type can only be queried or inserted in a table.

Syntax: RAW (precision)

Q56) What is meant by Joins? List out the types of joins.

Joins are used to extract data from multiple tables using some common column or condition.

There are various types of Joins as listed below:

- INNER JOIN
- OUTER JOIN

- CROSS JOINS or CARTESIAN PRODUCT
- EQUI JOIN
- ANTI JOIN
- SEMI JOIN

Q57) What is the difference between SUBSTR & INSTR functions?

SUBSTR function returns the sub-part identified by numeric values from the provided string.

Example: [Select SUBSTR ('India is my country', 1, 4) from dual] will return "Indi".

INSTR will return the position number of the sub-string within the string.

Example: [SELECT INSTR ('India is my country', 'a') from dual] will return 5.

Q58) How can we find out the duplicate values in an Oracle table?

We can use the below example query to fetch the duplicate records.

```
SELECT EMP_NAME, COUNT (EMP_NAME)
FROM EMP
GROUP BY EMP_NAME
HAVING COUNT (EMP_NAME) > 1;
```

Q59) How does the ON-DELETE-CASCADE statement work?

Using ON DELETE CASCADE will automatically delete a record in the child table when the same is deleted from the parent table. This statement can be used with Foreign Keys.
We can add ON DELETE CASCADE option on an existing table using the below set of commands.

Syntax:
ALTER TABLE CHILD_T1 ADD CONSTRAINT CHILD_PARENT_FK REFERENCES PARENT_T1 (COLUMN1) ON DELETE CASCADE;

Q60) What is a NVL function? How can it be used?
NVL is a function, which helps the user to substitute a value if null is encountered for an expression. It can be used as the below syntax.
[NVL (Value_In, Replace_With)]

Q61) What is the difference between a Primary Key & a Unique Key?
Primary key is used to identify each table row uniquely, while a Unique Key prevents duplicate values in a table column.
Given below are few differences:

- The primary key can be only one on the table while unique keys can be multiple.
- The primary key cannot hold null value at all while Unique key allows multiple null values.
- The primary key is a clustered index while a unique key is a non-clustered index.

Q62) How TRANSLATE command is different from REPLACE?

TRANSLATE command translates characters one by one in the provided string with the substitution character. REPLACE will replace a character or a set of characters with a complete substitution string.

Example:

TRANSLATE ('Missisippi','is','15) => M155151pp1

REPLACE ('Missisippi','is','15) => M15s15ippi

Q63) How can we find out the current date and time in Oracle?

We can find the current Date & Time using SYSDATE in Oracle.

Syntax:

SELECT SYSDATE into CURRENT_DATE from dual;

Q64) Why do we use COALESCE function in Oracle?

COALESCE function is used to return the first non-null expression from the list of arguments provided in the expression. Minimum two arguments should be there in an expression.

Syntax:

COALESCE (expr 1, expr 2, expr 3…expr n)

Q65) How will you write a query to get a 5th RANK student from a table STUDENT_REPORT?

The Query will be as follows:

```
SELECT TOP 1 RANK
FROM (SELECT TOP 5 RANK
FROM STUDENT_REPORT
```

ORDER BY RANK DESC) AS STUDENT
ORDER BY RANK ASC;

Q66) When do we use GROUP BY clause in a SQL Query?

GROUP BY clause is used to identify and group the data by one or more columns in the query results. This clause is often used with aggregate functions like COUNT, MAX, MIN, SUM, AVG etc.

Syntax:

```
SELECT COLUMN_1, COLUMN_2
FROM TABLENAME
WHERE [condition]
GROUP BY COLUMN_1, COLUMN_2
```

Q67) What is the quickest way to fetch the data from a table?

The quickest way to fetch the data would be to use ROWID in the SQL Query.

Q 68) Where do we use DECODE and CASE Statements?

Both DECODE & CASE statements will function like IF-THEN-ELSE statement and they are the alternatives for each other. These functions are used in Oracle to transform the data values.

Example:

DECODE Function

```
Select ORDERNUM,
DECODE (STATUS,'O', ‘ORDERED’,'P', ‘PACKED,’S’,’SHIPPED’,’A’,’ARRIVED’)
FROM ORDERS;
```

CASE Function

```
Select ORDERNUM
, CASE (WHEN STATUS ='O' then ‘ORDERED’
WHEN STATUS ='P' then PACKED
WHEN STATUS ='S' then ’SHIPPED’
ELSE ’ARRIVED’) END
FROM ORDERS;
```

Both the commands will display Order Numbers with respective Status as,

If,

Status O= Ordered

Status P= Packed

Status S= Shipped

Status A= Arrived

Q 69) Why do we need integrity constraints in a database?

Integrity constraints are required to enforce business rules so as to maintain the integrity of the database and prevent the entry of invalid data into the tables. With the help of the below-mentioned constraints, relationships can be maintained between the tables.

Various integrity constraints available include Primary Key, Foreign Key, UNIQUE KEY, NOT NULL & CHECK.

Q70) What do you mean by MERGE in Oracle and how can we merge two tables?

MERGE statement is used to merge the data from two tables. It selects the data from the source table and inserts/updates it in the other table based on the condition provided in the MERGE query.

Syntax:

```
MERGE                    INTO                    TARGET_TABLE_1
USING                                            SOURCE_TABLE_1
ON                                               SEARCH_CONDITION
WHEN                     MATCHED                 THEN
INSERT                   (COL_1,                 COL_2…)
VALUES                   (VAL_1,                 VAL_2…)
WHERE                                            <CONDITION>
WHEN          NOT                MATCHED         THEN
UPDATE        SET          COL_1=VAL_1,          COL_2=VAL_2…
WHEN <CONDITION>
```

Q71) What is the use of Aggregate functions in Oracle?

Aggregate functions perform summary operations on a set of values to provide a single value. There are several aggregate functions that we use in our code to perform calculations.

Few of them are listed below:

- AVG
- MIN
- MAX
- COUNT
- SUM
- STDEV

Q72) What are the set operators UNION, UNION ALL, MINUS & INTERSECT meant to do?

Set operator facilitates the user to fetch the data from two or more than two tables at once if the columns and relative data types are same in the source tables.

- UNION operator returns all the rows from both the tables except the duplicate rows.
- UNION ALL returns all the rows from both the tables along with the duplicate rows.
- MINUS returns rows from the first table, which does not exist in the second table.
- INTERSECT returns only the common rows in both the tables.

Q73) Can we convert a date to char in Oracle and if so, what would be the syntax?

We can use the TO_CHAR function to do the above conversion.

The syntax will be as follows:

[SELECT to_char (to_date ('30-01-2018', 'DD-MM-YYYY'), 'YYYY-MM-DD') FROM dual;]

Q 74) What do you mean by a database transaction & what all TCL statements are available in Oracle?

Transaction occurs when a set of SQL statements are executed in one go. To control the execution of these statements, Oracle has introduced TCL i.e. Transaction Control Statements that use a set of statements.

The set of statements include:

- COMMIT: Used to make a transaction permanent.
- ROLLBACK: Used to roll back the state of DB to last the commit point.
- SAVEPOINT: Helps to specify a transaction point to which rollback can be done later.

Q75) What do you understand by a database object? Can you list a few of them?

An object used to store the data or references of the data in a database is known as a Database object. The database consists of various types of DB objects such as tables, views, indexes, constraints, stored procedures, triggers etc.

Q76) What is a Nested table and how is it different from a normal table?

A nested table is a database collection object, which can be stored as a column in a table. While creating a normal table, an entire nested table can be referenced in a single column. Nested tables have only one column with no restriction of rows.

Example:

```
CREATE TABLE EMP (
EMP_ID NUMBER,
EMP_NAME  TYPE_NAME)
```

Here we are creating a normal table as EMP and referring a nested table TYPE_NAME as a column.

Q77) Can we save images in a database and if yes, how?

BLOB stands for Binary Large Object, which is a datatype that is generally used to hold images, audio & video files or some binary executables.

This datatype has the capacity of holding data up to 4 GB.

Q78) What do you understand by database schema and what does it hold?

Schema is a collection of database objects owned by a database user who can create or manipulate new objects within this schema.

The schema can contain any DB objects like table, view, indexes, clusters, stored procs, functions etc.

Q79) What is a Data Dictionary and how can it be created?

Whenever a new database is created, a database specific data dictionary gets created by the system. This dictionary maintains all the metadata related to the database and owned by the SYS user.It has a set of read-only tables and views and it is physically stored in the SYSTEM tablespace.

Q80) What is a View and how is it different from a table?

A view is a user-defined database object that is used to store the results of a SQL query, which can be referenced later. Views do not store this data physically but as a virtual table, hence it can be referred as a logical table.

A table can hold data but not SQL Query results whereas View can save the query results, which can be used in another SQL Query as a whole.The table can be updated or deleted while Views cannot be done so.

Q81) What is meant by a deadlock situation?

Deadlock is a situation when two or more users are simultaneously waiting for the data, which is locked by each other and hence, results in all blocked user sessions.

Q82) What is meant by an index?

An index is a schema object, which is created to search the data efficiently within the table. Indexes are usually created on certain columns of the table, which are accessed the most.

Indexes can be clustered or non-clustered.

Q83) What is a Role in Oracle database?

Giving access to individual objects to the individual users is a tough administrative task. In order to make this job easy, a group of common privileges is created in a database, which is known as Role. The role, once created can be assigned to or revoked from the users by using Grant & Revoke command.

Syntax:

```
CREATE ROLE READ_TABLE_ROLE;
GRANT SELECT ON EMP TO READ_TABLE_ROLE;
GRANT READ_TABLE_ROLE TO USER1;
REVOKE READ_TABLE_ROLE FROM USER1;
```

Q84) What are the attributes that are found in a cursor?

A cursor has various attributes as mentioned below:

(i) %FOUND:

- Returns INVALID_CURSOR if the cursor has been declared but closed.
- Returns NULL if fetch has not happened but the cursor is open only.
- Returns TRUE if the rows are fetched successfully and FALSE if no rows are returned.

(ii) NOT FOUND:

- Returns INVALID_CURSOR if the cursor has been declared but closed.
- Returns NULL if fetch has not happened but the cursor is open only.
- Returns FALSE if rows are fetched successfully and TRUE if no rows are returned

(iii) %ISOPEN: Returns TRUE if the cursor is OPEN else FALSE

(iv) %ROWCOUNT: Returns the count of fetched rows.

Q85) Why do we use %ROWTYPE & %TYPE in PLSQL?

%ROWTYPE & %TYPE are the attributes in PL/SQL which can inherit the datatypes of a table defined in a database. The purpose of using these attributes is to provide data independence and integrity.If

any of the datatypes or precision gets changed in the database, PL/SQL code gets updated automatically with the data type changes.

%TYPE is used for declaring a variable which needs to have the same data type as of a table column.While %ROWTYPE will be used to define a complete row of record having a structure similar to the structure of a table.

Q86) Why do we create Stored Procedures & Functions in PL/SQL and how are they different?

A stored procedure is a set of SQL statements that are written to perform a specific task. These statements can be saved as a group in the database with an assigned name and can be shared with different programs if permissions are there to access the same.

Functions are again subprograms that are written to perform specific tasks but there are differences between both of them.

Stored Procedures	Functions
SPs may or may not return a value and can return multiple values as well.	Function will always return only single value.
SPs can include DML statements like insert, update & delete.	We cannot use DML statements in a function.

Stored Procedures	Functions
SPs can call functions.	Functions cannot call stored procedures.
SPs support exception handling using Try/Catch block.	Functions does not support Try/Catch block.

Q 87) What are the parameters that we can pass through a stored procedure?

We can pass IN, OUT & INOUT parameters through a stored procedure and they should be defined while declaring the procedure itself.

Q88) What is a trigger and what are its types?

A trigger is a stored program which is written in such a way that it gets executed automatically when some event occurs. This event can be any DML or a DDL operation.

PL/SQL supports two types of triggers:

- Row Level
- Statement Level

Q89) How will you distinguish a global variable with a local variable in PL/SQL?

Global variable is the one, which is defined at the beginning of the program and survives until the end.It can be accessed by any methods or procedures within the program, while the access to the local variable is limited to the procedure or method where it is declared.

Q90) What are the packages in PL SQL?

A Package is a group of related database objects like stored procs, functions, types, triggers, cursors etc. that are stored in Oracle database. It is a kind of library of related objects which can be accessed by multiple applications if permitted.

PL/SQL Package structure consists of 2 parts: package specification & package body

Q91)What is a Self-Join?

A self JOIN is a case of regular join where a table is joined to itself based on some relation between its own column(s). Self-join uses the INNER JOIN or LEFT JOIN clause and a table alias is used to assign different names to the table within the query.

```
SELECT A.emp_id AS"Emp_ID",A.emp_name AS"Employee",
B.emp_id AS"Sup_ID",B.emp_name AS"Supervisor"
FROM employee A, employee B
WHERE A.emp_sup = B.emp_id;
```

Q92)What is a Cross-Join?

Cross join can be defined as a cartesian product of the two tables included in the join. The table after join contains the same number of rows as in the cross-product of number of rows in the two tables. If a WHERE clause is used in cross join then the query will work like an INNER JOIN.

```
SELECT stu.name, sub.subject
FROM students AS stu
CROSSJOIN subjects AS sub;
```

Q93)What is Data Integrity?

Data Integrity is the assurance of accuracy and consistency of data over its entire life-cycle, and is a critical aspect to the design, implementation and usage of any system which stores, processes, or retrieves data. It also defines integrity constraints to enforce business rules on the data when it is entered into an application or a database.

Q94)What are UNION, MINUS and INTERSECT commands?

The UNION operator combines and returns the result-set retrieved by two or more SELECT statements. The MINUS operator in SQL is used to remove duplicates from the result-set obtained by the second SELECT query from the result-set obtained by the first SELECT query and then return the filtered results from the first.

The INTERSECT clause in SQL combines the result-set fetched by the two SELECT statements where records from one match the other and then returns this intersection of result-sets.
Certain conditions need to be met before executing either of the above statements in SQL -
Each SELECT statement within the clause must have the same number of columns
The columns must also have similar data types
The columns in each SELECT statement should necessarily have the same order

Q95) What is an Alias in SQL?
An alias is a feature of SQL that is supported by most, if not all, RDBMSs. It is a temporary name assigned to the table or table column for the purpose of a particular SQL query. In addition, aliasing can be employed as an obfuscation technique to secure the real names of database fields. A table alias is also called a correlation name .
An alias is represented explicitly by the AS keyword but in some cases the same can be performed without it as well. Nevertheless, using the AS keyword is always a good practice.

```
SELECT A.emp_name AS "Employee"      /* Alias using AS keyword */
B.emp_name AS "Supervisor"
FROM employee A, employee B          /* Alias without AS keyword */
WHERE A.emp_sup = B.emp_id;
```

Q96) What is Collation? What are the different types of Collation Sensitivity?

Collation refers to a set of rules that determine how data is sorted and compared. Rules defining the correct character sequence are used to sort the character data. It incorporates options for specifying case-sensitivity, accent marks, kana character types and character width. Below are the different types of collation sensitivity:

- Case sensitivity: A and a are treated differently.
- Accent sensitivity: a and á are treated differently.
- Kana sensitivity: Japanese kana characters Hiragana and Katakana are treated differently.
- Width sensitivity: Same character represented in single-byte (half-width) and double-byte (full-width) are treated differently.

Q97) What is a Stored Procedure?

A stored procedure is a subroutine available to applications that access a relational database management system (RDBMS). Such procedures are stored in the database data dictionary. The sole disadvantage of stored procedure is that it can be executed nowhere except in the database and occupies more memory in the database server. It also provides a sense of security and functionality as users who can't access the data directly can be granted access via stored procedures.

```
DELIMITER$$
CREATEPROCEDURE FetchAllStudents()
BEGIN
SELECT*FROM myDB.students;
```

```
END$$
DELIMITER ;
```

Q98)How to create empty tables with the same structure as another table?

Creating empty tables with the same structure can be done smartly by fetching the records of one table into a new table using the INTO operator while fixing a WHERE clause to be false for all records. Hence, SQL prepares the new table with a duplicate structure to accept the fetched records but since no records get fetched due to the WHERE clause in action, nothing is inserted into the new table.

```
SELECT*INTO Students_copy
FROM Students WHERE1 = 2;
```

Q.99)Write a SQL query to fetch employee names having a salary greater than or equal to 5000 and less than or equal 10000.

Ans. Here, we will use BETWEEN in the 'where' clause to return the EmpId of the employees with salary satisfying the required criteria and then use it as subquery to find the fullName of the employee form EmployeeDetails table.

```
    SELECTFullName
    FROMEmployeeDetails
```

```
WHEREEmpId IN
(SELECTEmpId FROMEmpolyeeSalary
WHERESalary BETWEEN5000 AND10000);
```

Q100). Write a SQL query to fetch project-wise count of employees sorted by project's count in descending order.

Ans. The query has two requirements – first to fetch the project-wise count and then to sort the result by that count. For project-wise count, we will be using GROUPBY clause and for sorting, we will use ORDER BY clause on the alias of the project-count.

```
SELECTProject, count(EmpId) EmpProjectCount
FROMEmployeeSalary
GROUP BY Project
ORDERBYEmpProjectCount DESC;
```