Q1. What are the basic Datatypes supported in C Programming Language?

Ans: The Datatypes in C Language are broadly classified into 4 categories. They are as follows:

- Basic Datatypes
- Derived Datatypes
- Enumerated Datatypes
- Void Datatypes

The Basic Datatypes supported in C Language are as follows:

Datatype Name	Datatype Size	Datatype Range
short	**1 byte**	-128 to 127
unsigned short	**1 byte**	0 to 255
char	**1 byte**	-128 to 127
unsigned char	**1 byte**	0 to 255
int	**2 bytes**	-32,768 to 32,767
unsigned int	**2 bytes**	0 to 65,535
long	**4 bytes**	-2,147,483,648 to 2,147,483,647
unsigned long	**4 bytes**	0 to 4,294,967,295
float	**4 bytes**	3.4E-38 to 3.4E+38
double	**8 bytes**	1.7E-308 to 1.7E+308
long double	**10 bytes**	3.4E-4932 to 1.1E+4932

Q2. What do you mean by Dangling Pointer Variable in C Programming?

Ans: A **Pointer** in C Programming is used to point the memory location of an existing variable. In case if that particular variable is deleted and the Pointer is still pointing to the same memory location, then that particular pointer variable is called as a **Dangling Pointer Variable.**

Q3. What do you mean by the Scope of the variable? What is the scope of the variables in C?

Ans: **Scope** of the variable can be defined as the part of the code area where the variables declared in the program can be accessed directly. In C, all identifiers are lexically (or statically) scoped.

Q4. What are static variables and functions?

Ans: The variables and functions that are declared using the keyword **Static_** are considered as Static Variable and Static Functions. The variables declared using Static keyword will have their scope restricted to the function in which they are declared.

Q5. Differentiate between calloc() and malloc()

Ans: **calloc()** and **malloc()** are memory dynamic memory allocating functions. The only difference between them is that calloc() will load all the assigned memory locations with value 0 but malloc() will not.

Q6. What are the valid places where the programmer can apply Break Control Statement?

Ans: Break Control statement is valid to be used inside a loop and **Switch control statements.**

Q7. How can we store a negative integer?

Ans: To store a negative integer, we need to follow the following steps. Calculate the two's complement of the same positive integer.

Eg: 1011 (-5)

Step-1 − One's complement of 5: 1010

Step-2 − Add 1 to above, giving 1011, which is -5

Q8. Differentiate between Actual Parameters and Formal Parameters.

Ans: The Parameters which are sent from main function to the subdivided function are called as **Actual Parameters** and the parameters which are declared a the Subdivided function end are called as **Formal Parameters.**

Q9. Can a C program be compiled or executed in the absence of a main()?

Ans: The program will be compiled but will not be executed. To execute any C program, main() is required.

Q10. What do you mean by a Nested Structure?

Ans: When a data member of one structure is referred by the data member of another function, then the structure is called a **Nested Structure.**

Q11. What is a C Token?

Ans: *Keywords, Constants, Special Symbols, Strings, Operators, Identifiers* used in C program are referred to as **C Tokens.**

Q12. What is Preprocessor?

Ans: A Preprocessor Directive is considered as a built-in predefined function or macro that acts as a directive to the compiler and it gets executed before the actual C Program is executed.

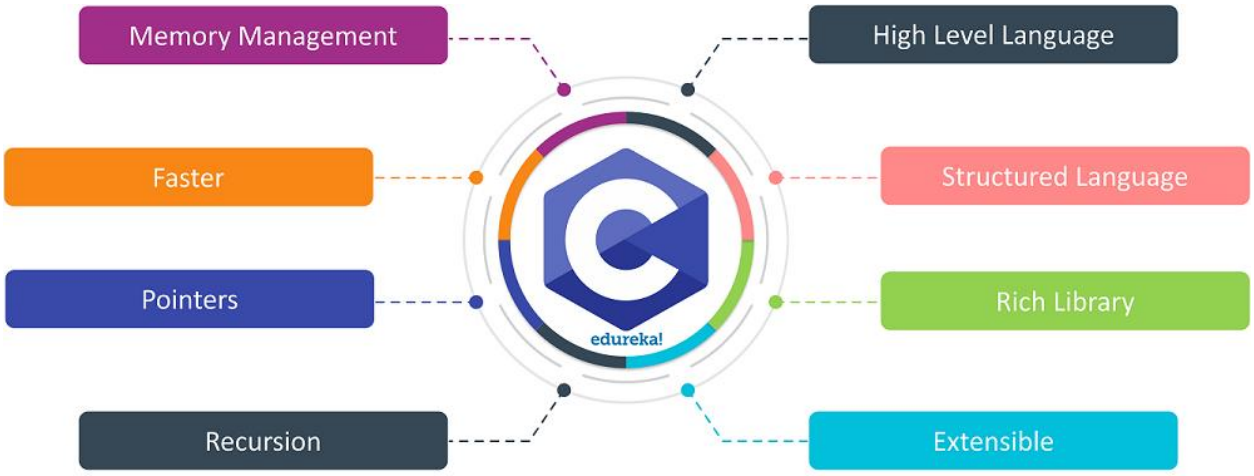
In case you are facing any challenges with these C Programming Interview Questions, please write your problems in the comment section below.

Q13. Why is C called the Mother of all Languages?

Ans: C introduced many core concepts and data structures like **arrays, lists, functions, strings,** etc. Many languages designed after C are designed on the basis of C Language. Hence, it is considered as the mother of all languages.

Q14. Mention the features of C Programming Language.

Ans:



Q15. What is the purpose of printf() and scanf() in C Program?

Ans: printf() is used to print the values on the screen. To print certain values, and on the other hand, **scanf()** is used to scan the values. We need an appropriate datatype format specifier for both printing and scanning purposes. For example,

- **%d**: It is a datatype format specifier used to print and scan an **integer** value.
- **%s**: It is a datatype format specifier used to print and **scan** a string.
- **%c**: It is a datatype format specifier used to display and scan a **character** value.
- **%f**: It is a datatype format specifier used to display and scan a **float** value.

Q16. What is an array?

Ans. The array is a simple data structure that stores multiple elements of the same datatype in a reserved and sequential manner. There are three types of arrays, namely,

- One Dimensional Array
- Two Dimensional Array
- Multi-Dimensional Array

Q17. What is /0 character?

Ans: The Symbol mentioned is called a **Null Character**. It is considered as the terminating character used in strings to notify the end of the string to the compiler.

Q18. What is the main difference between the Compiler and the Interpreter?

Ans: Compiler is used in C Language and it translates the complete code into the Machine Code in one shot. On the other hand, Interpreter is used in Java Programming Langauge and other high-end programming languages. It is designed to compile code in line by line fashion.

Q19. Can I use int datatype to store 32768 value?

Ans: No, Integer datatype will support the range between **-32768 and 32767.** Any value exceeding that will not be stored. We can either use **float** or **long int.**

Intermediate C Programming Interview Questions

Q20. How is a Function declared in C Language?

Ans: A function in C language is declared as follows,

```
return_type function_name(formal parameter list)
                          {
                        Function_Body;
                          }
```

Q21. What is Dynamic Memory allocation? Mention the syntax.

Ans: Dynamic Memory Allocation is the process of allocating memory to the program and its variables in runtime. Dynamic Memory Allocation process involves three functions for allocating memory and one function to free the used memory.

malloc() – Allocates memory

Syntax:

```
ptr = (cast-type*) malloc(byte-size);
```

calloc() – Allocates memory

Syntax:

```
ptr = (cast-type*)calloc(n, element-size);
```

realloc() – Allocates memory

Syntax:

```
ptr = realloc(ptr, newsize);
```

free() – Deallocates the used memory

Syntax:

```
free(ptr);
```

Q22. What do you mean by Dangling Pointer Variable in C Programming?

Ans: A **Pointer** in C Programming is used to point the memory location of an existing variable. In case if that particular variable is deleted and the Pointer is still pointing to the same memory location, then that particular pointer variable is called as a **Dangling Pointer Variable.**

Q23. Where can we not use &(address operator in C)?

Ans: We cannot use **&** on **constants** and on a variable which is declared using the **register storage** class.

Q24. Write a simple example of a structure in C Language

Ans: Structure is defined as a user-defined data type that is designed to store multiple data members of the different data types as a single unit. A structure will consume the memory equal to the summation of all the data members.

```
1
2                                   struct employee
3                                          {
4                                      char name[10];
5                                         int age;
6                                         }e1;
7                                      int main()
8                                          {
9                                   printf("Enter the name");
10                                     scanf("%s",e1.name);
11                                      printf("n");
12                                  printf("Enter the age");
13                                    scanf("%d",&e1.age);
14                                      printf("n");
15          printf("Name and age of the employee: %s,%d",e1.name,e1.age);
16                                    return 0;
                                        }
```

Q25. Differentiate between call by value and call by reference.

Ans:

Factor	Call by Value	Call by Reference
Safety	Actual arguments cannot be changed and remain safe	Operations are performed on actual arguments, hence not safe
Memory Location	Separate memory locations are created for actual and formal arguments	Actual and Formal arguments share the same memory space.
Arguments	Copy of actual arguments are sent	Actual arguments are passed

//Example of Call by Value method

```
#include<stdio.h>
void change(int,int);
int main()
{
int a=25,b=50;
change(a,b);
printf("The value assigned to a is: %d",a);
printf("n");
printf("The value assigned to of b is: %d",b);
return 0;
}
void change(int x,int y)
{
x=100;
y=200;
}
```

//Output

```
The value assigned to of a is: 25
The value assigned to of b is: 50
```

//Example of Call by Reference method

```
#include<stdio.h>
void change(int*,int*);
int main()
{
int a=25,b=50;
change(&a,&b);
printf("The value assigned to a is: %d",a);
printf("n");
printf("The value assigned to b is: %d",b);
return 0;
}
void change(int *x,int *y)
{
*x=100;
*y=200;
}
```

//Output

```
The value assigned to a is: 100
The value assigned to b is: 200
```

In case you are facing any challenges with these C Programming Interview Questions, please write your problems in the comment section below.

Q26. Differentiate between getch() and getche().

Ans: Both the functions are designed to read characters from the keyboard and the only difference is that

getch(): reads characters from the keyboard but it does not use any buffers. Hence, data is not displayed on the screen.

getche(): reads characters from the keyboard and it uses a buffer. Hence, data is displayed on the screen.

//Example

```
#include<stdio.h>
#include<conio.h>
int main()
{
char ch;
printf("Please enter a character ");
ch=getch();
printf("nYour entered character is %c",ch);
printf("nPlease enter another character ");
```

```
                                        ch=getche();
                                printf("nYour new character is %c",ch);
                                        return 0;
                                         }
```

//Output

```
Please enter a character
Your entered character is x
Please enter another character z
Your new character is z
```

Q27. Explain toupper() with an example.

Ans. toupper() is a function designed to convert lowercase words/characters into upper case.

//Example

```
                                #include<stdio.h>
                                #include<ctype.h>
                                    int main()
                                        {
                                       char c;
                                        c=a;
                        printf("%c after conversions  %c", c, toupper(c));
                                        c=B;
                        printf("%c after conversions  %c", c, toupper(c));
```

//Output:

```
a after conversions A
B after conversions B
```

Q28. Write a code to generate random numbers in C Language.

Ans: Random numbers in C Language can be generated as follows:

```
                                #include<stdio.h>
                                #include<stdlib.h>
                                    int main()
                                        {
                                       int a,b;
                                  for(a=1;a<=10;a++)
```

```
6                                                   {
7                                                  b=rand();
8                                                printf("%dn",b);
9                                                   }
10                                               return 0;
11                                                }
12
```

//Output

```
1987384758
2057844389
3475398489
2247357398
1435983905
```

Q29. Can I create a customized Head File in C language?

Ans: It is possible to create a new header file. Create a file with function prototypes that need to be used in the program. Include the file in the '#include' section in its name.

Q30. What do you mean by Memory Leak?

Ans: Memory Leak can be defined as a situation where programmer allocates dynamic memory to the program but fails to free or delete the used memory after the completion of the code. This is harmful if daemons and servers are included in the program.

```
1                                                #include<stdio.h>
2                                               #include<stdlib.h>
3                                                   int main()
4                                                       {
5                                                    int* ptr;
6                                                  int n, i, sum = 0;
7                                                      n = 5;
8                                  printf("Enter the number of elements: %dn", n);
9                                        ptr = (int*)malloc(n * sizeof(int));
10                                                  if (ptr == NULL)
11                                                      {
12                                              printf("Memory not allocated.n");
13                                                   exit(0);
14                                                   }
15                                                  else
16                                                   {
17                                 printf("Memory successfully allocated using malloc.n");
                                                  for (i = 0; i<= n; ++i)
                                                       {
                                                    ptr[i] = i + 1;
```

```
                }
printf("The elements of the array are: ");
            for (i = 0; i<=n; ++i)
                {
              printf("%d, ", ptr[i]);
                }
            }
          return 0;
            }
```

//Output

```
Enter the number of elements: 5
Memory successfully allocated using malloc.
The elements of the array are: 1, 2, 3, 4, 5,
```

In case you are facing any challenges with these C Programming Interview Questions, please write your problems in the comment section below.

Q31. Explain Local Static Variables and what is their use?

Ans: A local static variable is a variable whose life doesn't end with a function call where it is declared. It extends for the lifetime of the complete program. All calls to the function share the same copy of local static variables.

```

#include<stdio.h>
    void fun()
        {
     static int x;
  printf("%d ", x);
      x = x + 1;
        }
    int main()
        {
       fun();
       fun();
     return 0;
        }

```

//Output

```
0 1
```

Q32. What is the difference between declaring a header file with < > and ” “?

Ans: If the Header File is declared using < > then the compiler searches for the header file within the Built-in Path. If the Header File is declared using ” ” then the compiler will search for the Header File in the current working directory and if not found then it searches for the file in other locations.

Q33. When should we use the register storage specifier?

Ans: We use Register Storage Specifier if a certain variable is used very frequently. This helps the compiler to locate the variable as the variable will be declared in one of the CPU registers.

Q34. Which statement is efficient and why? x=x+1; or x++; ?

Ans: x++; is the most efficient statement as it just a single instruction to the compiler while the other is not.

Q35. Can I declare the same variable name to the variables which have different scopes?

Ans: Yes, Same variable name can be declared to the variables with different variable scopes as the following example.

```
1
2                                   int var;
3                               void function()
4                                   {
5                                int variable;
6                                   }
7                               int main()
8                                   {
9                                int variable;
                                    }
```

Q36. Which variable can be used to access Union data members if the Union variable is declared as a pointer variable?

Ans: Arrow Operator(->) can be used to access the data members of a Union if the Union Variable is declared as a pointer variable.

Q37. Mention File operations in C Language.

Ans: **Basic File Handling Techniques in C**, provide the basic functionalities that user can perform against files in the system.

Function	Operation
fopen()	**To Open a File**
fclose()	**To Close a File**
fgets()	**To Read a File**
fprint()	**To Write into a File**

In case you are facing any challenges with these C Programming Interview Questions, please write your problems in the comment section below.

Q38. What are the different storage class specifiers in C?

Ans: The different storage specifiers available in C Language are as follows:

- **auto**
- **register**
- **static**
- **extern**

Q39. What is typecasting?

Ans: 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;
```

Q40. Write a C program to print hello world without using a semicolon (;).

Ans:

```
#include<stdio.h>
void main()
{
if(printf("hello world")){}
}
```

//Output:

```
hello world
```

Q41. Write a program to swap two numbers without using the third variable.

Ans:

```
#include<stdio.h>
#include<conio.h>
main()
{
int a=10, b=20;
clrscr();
printf("Before swapping a=%d b=%d",a,b);
a=a+b;
b=a-b;
a=a-b;
printf("nAfter swapping a=%d b=%d",a,b);
getch();
}
```

//Output

```
Before swapping a=10 b=20
After swapping a=20 b=10
```

Advanced C Programming Interview Questions

Q42. How can you print a string with the symbol % in it?

Ans: There is no escape sequence provided for the symbol % in C. So, to print % we should use '%%' as shown below.

```
printf("there are 90%% chances of rain tonight");
```

Q43. Write a code to print the following pattern.

```
1
12
123
1234
12345
```

Ans: To print the above pattern, the following code can be used.

```
#include<stdio.h>
int main()
{
for(i=1;i<=5;1++)
{
for(j=1;j<=5;j++)
{
print("%d",j);
}
printf("n");
}
return 0;
}
```

Q44. Explain the # pragma directive.

Ans: The following points explain the Pragma Directive.

- This is a preprocessor directive that can be used to turn on or off certain features.
- It is of two types #pragma startup, #pragma exit and pragma warn.

- #pragma startup allows us to specify functions called upon program startup.
- #pragma exit allows us to specify functions called upon program exit.
- #pragma warn tells the computer to suppress any warning or not.

Q45. How can you remove duplicates in an array?

Ans: The following program will help you to remove duplicates from an array.

```
#include <stdio.h>
int main()
{
int n, a[100], b[100], calc = 0, i, j,count;
printf("Enter no. of elements in array.n");
scanf("%d", &n);
printf("Enter %d integersn", n);
for (i = 0; i < n; i++)
scanf("%d", &a[i]);
for (i = 0; i<n; i++)
{
for (j = 0; j<calc; j++)
{
if(a[i] == b[j])
break;
}
if (j== calc)
{
b[count] = a[i];
calc++;
}
}
printf("Array obtained after removing duplicate elementsn");
for (i = 0; i<calc; i++)
{
printf("%dn", b[i]);
}
return 0;
}
```

//Output

```
Enter no. of elements in array. 5
Enter 5 integers
12
11
11
10
```

```
4
Array obtained after removing duplicate elements
12
11
10
4
```

Q46. What is Bubble Sort Algorithm? Explain with a program.

Ans: Bubble sort is a simple sorting algorithm that repeatedly steps through the list, compares adjacent elements and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted.

The following code executes Bubble Sort.

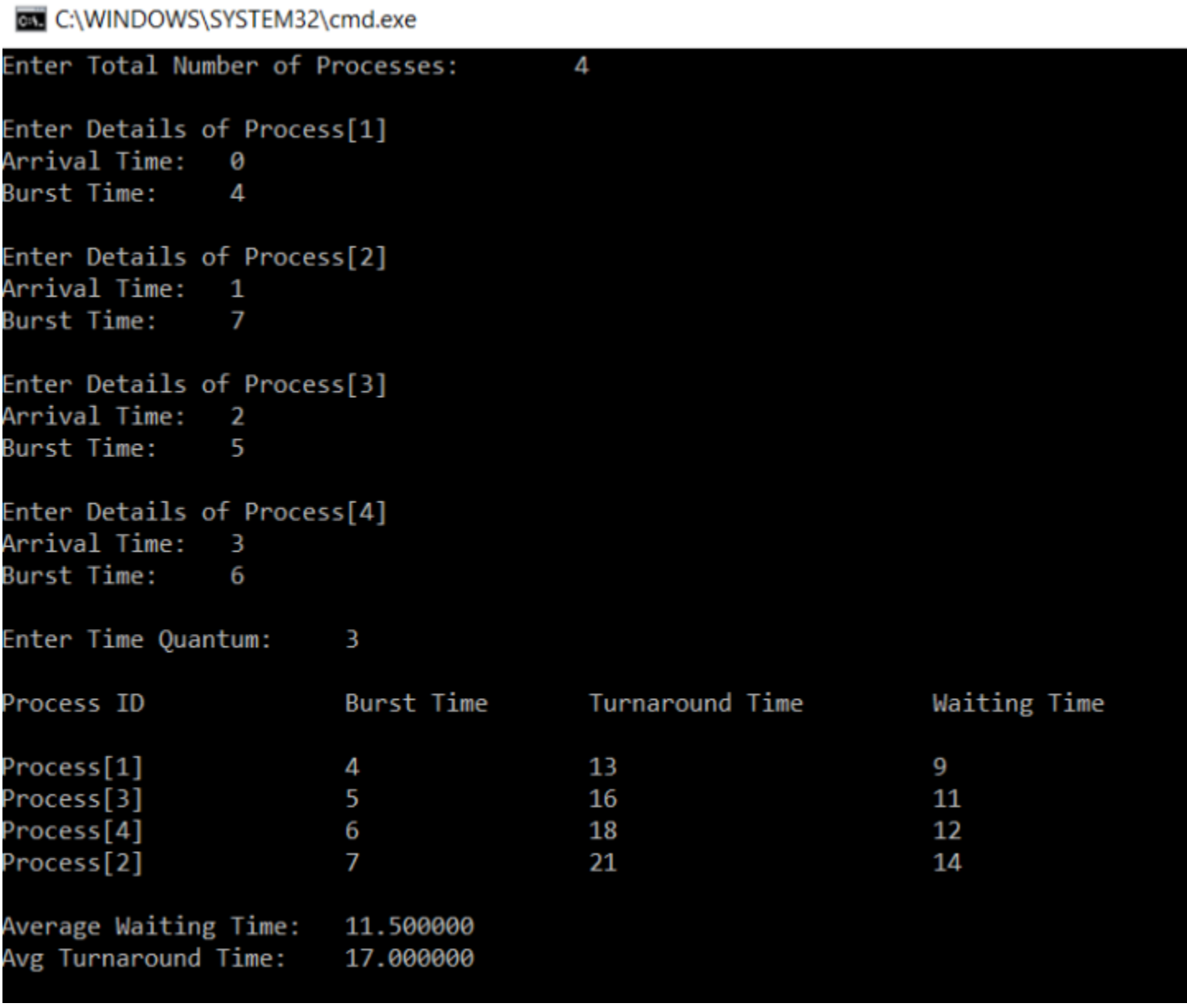
```
1
2
3                                   int main()
4                                       {
5                                 int array[100], n, i, j, swap;
6                               printf("Enter number of elementsn");
7                                       scanf("%d", &n);
8                                 printf("Enter %d Numbers:n", n);
9                                     for(i = 0; i<n; i++)
10                                     scanf("%d", &array[i]);
11                                    for(i = 0 ; i<n - 1; i++)
12                                                {
13                  for(j = 0 ; j < n-i-1; j++) { if(array[j]>array[j+1])
14                                                    {
15                                                swap=array[j];
16                                              array[j]=array[j+1];
17                                                array[j+1]=swap;
18                                                    }
19                                                 }
20                                               }
21                                    printf("Sorted Array:n");
22                                      for(i = 0; i < n; i++)
23                                        printf("%dn", array[i]);
                                              return 0;
                                          }
```

Q47. What is Round-robin algorithm? Write a code for Round Robin Scheduling.

Ans: Round-robin Algorithm is one of the algorithms employed by process and network schedulers in computing in order to evenly distribute resources in the system.

The following code will execute Round Robin Scheduling

//Output



In case you are facing any challenges with these C Programming Interview Questions, please write your problems in the comment section below.

Q48. Which structure is used to link the program and the operating system?

```
#include<stdio.h>

int main()
{
int i, limit, total = 0, x, counter = 0, time_quantu
int wait_time = 0, turnaround_time = 0, arrival_time[10], burst_tim
float average_wait_time, average_turnaround_time;
printf("nEnter Total Number of Processes:t");
scanf("%d", &limit);
x = limit;
for(i = 0; i<limit; i++)
{
printf("nEnter Details of Process[%d]n", i + 1);
printf("Arrival Time:t");
scanf("%d", &arrival_time[i]);
printf("Burst Time:t");
scanf("%d", &burst_time[i]);
temp[i] = burst_time[i];
}

printf("nEnter Time Quantum:t");
scanf("%d", &time_quantum);
printf("nProcess IDttBurst Timet Turnaround Timet Waiting
for(total = 0, i = 0; x != 0;)
{
if(temp[i] <= time_quantum && temp[i] > 0)
{
total = total + temp[i];
temp[i] = 0;
counter = 1;
}
else if(temp[i]>0)
{
temp[i] = temp[i] - time_quantum;
total = total + time_quantum;
}
if(temp[i] == 0 && counter == 1)
{
x--;
printf("nProcess[%d]tt%dtt %dttt %d", i + 1, burst_time[i], total - arrival_time[i], tot
wait time = wait time + total - arrival time[i] - burst t
turnaround_time = turnaround_time + total - arrival_tim
counter = 0;
}
if(i == limit - 1)
{
i = 0;            }

else if(arrival_time[i + 1] <= total)
```

```
                                                    {
                                                  i++;
                                                    }
                                                  else
                                                    {
                                                 i = 0;
                                                    }
                                                    }


                                        average_wait_time = wait_time * 1.0 / limit;
                                    average_turnaround_time = turnaround_time * 1.0 / limi
                                     printf("nnAverage Waiting Time:t%f", average_wait_tim
                                printf("nAvg Turnaround Time:t%fn", average_turnaround_t
                                                    return 0;
}
```

Ans: The answer can be explained through the following points,

- The structure used to link the operating system to a program is **file.**
- The **file** is defined in the header file **“stdio.h”(standard input/output** header file).
- It contains the information about the file being used, its current **size** and its **location** in memory.
- It contains a character **pointer** that points to the **character** that is being opened.
- Opening a file establishes a **link** between the **program** and the **operating system** about which file is to be accessed.

Q49. What are the limitations of scanf() and how can it be avoided?

Ans: The Limitations of scanf() are as follows:

- **scanf()** cannot work with the string of characters.
- It is not possible to enter a **multiword** string into a **single** variable using scanf().
- To avoid this the **gets(**) function is used.
- It gets a string from the keyboard and is terminated when **enter** key is pressed.

- Here the **spaces** and **tabs** are acceptable as part of the input string.

Q50. Differentiate between the macros and the functions.

Ans: The differences between macros and functions can be explained as follows:

- **Macro** call replaces the templates with the expansion in a literal way.
- The **Macro** call makes the program run **faster** but also increases the program size.
- Macro is **simple** and avoids **errors** related to the function calls.
- In a function, call **control** is transferred to the function along with arguments.
- It makes the functions **small** and **compact.**
- Passing **arguments** and getting back the **returned value** takes time and makes the program run at a **slower rate.**

Q51. Suppose a global variable and local variable have the same name. Is it is possible to access a global variable from a block where local variables are defined?

Ans: No. It is not possible in C. It is always the most local variable that gets preference.

With this, we come to an end of this “C Programming Interview Questions” article. I hope you have understood the importance of C Programming.