CAMPUSSUTRAA

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.

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

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.

4) Who is the founder of C language?

Dennis Ritchie.

5) When was C language developed?

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

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.

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.

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.

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.

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:

```c
#include <stdio.h>
void change(int,int);
int main()
{
    int a=10,b=20;
    change(a,b); //calling a function by passing the values of variables.
    printf("Value of a is: %d",a);
    printf("\n");
    printf("Value of b is: %d",b);
    return 0;
}
void change(int x,int y)
{
    x=13;
    y=17;
}
```

Output:

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

Example of call by reference:

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

```c
void change(int*,int*);
int main()
{
    int a=10,b=20;
    change(&a,&b); // calling a function by passing references of variables.
    printf("Value of a is: %d",a);
    printf("\n");
    printf("Value of b is: %d",b);
    return 0;
}
void change(int *x,int *y)
{
    *x=13;
    *y=17;
}
```

Output:

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

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:

33. Winding phase
34. 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

```c
#include <stdio.h>
int calculate_fact(int);
int main()
```

```
{
 int n=5,f;
 f=calculate_fact(n); // calling a function
 printf("factorial of a number is %d",f);
  return 0;
}
int calculate_fact(int a)
{
  if(a==1)
  {
      return 1;
  }
  else
  return a*calculate_fact(a-1); //calling a function recursively.
   }
```

Output:

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

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:

```
data_type array_name[size];
```

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

Syntax:

```
data_type array_name[size];
```

Example of an array:

```
#include <stdio.h>
int main()
{
  int arr[5]={1,2,3,4,5}; //an array consists of five integer values.
  for(int i=0;i<5;i++)
  {
      printf("%d ",arr[i]);
  }
   return 0;
}
```

Output:

```
1 2 3 4 5
```

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:

```
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

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

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.

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.

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

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:

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

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:

```c
#include <stdio.h>
 int main()
{
   int a=10;
   int *ptr,**pptr; // *ptr is a pointer and **pptr is a double pointer.
   ptr=&a;
   pptr=&ptr;
   printf("value of a is:%d",a);
   printf("\n");
   printf("value of *ptr is : %d",*ptr);
   printf("\n");
   printf("value of **pptr is : %d",**pptr);
   return 0;
}
```

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.

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.

102. For example:
103. `int a[10];`

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

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.

104. For example
105. `int *p= malloc(sizeof(int)*10);`

The above example allocates the memory at runtime.

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

106. 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**

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

107. calloc()

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

- **Syntax**

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

108. 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**

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

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

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

111. Syntax

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

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

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.

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

```
struct structure_name
{
  Member_variable1;
 Member_variable2
.
.
}[structure variables];
```

Let's see a simple example.

```
#include <stdio.h>
struct student
{
   char name[10];       // structure members declaration.
   int age;
}s1;      //structure variable
int main()
{
   printf("Enter the name");
   scanf("%s",s1.name);
   printf("\n");
   printf("Enter the age");
   scanf("%d",&s1.age);
   printf("\n");
   printf("Name and age of a student: %s,%d",s1.name,s1.age);
   return 0;
}
```

Output:

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

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

```
union union_name
{
Member_variable1;
Member_variable2;
.
.
Member_variable n;
}[union variables];
```

Let's see a simple example

```
#include<stdio.h>
union data
{
    int a;      //union members declaration.
    float b;
    char ch;
};
int main()
{
  union data d;      //union variable.
  d.a=3;
  d.b=5.6;
  d.ch='a';
```

```c
printf("value of a is %d",d.a);
printf("\n");
printf("value of b is %f",d.b);
printf("\n");
printf("value of ch is %c",d.ch);
return 0;
}
```

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.

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

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

Let's see a simple example

```c
#include<stdio.h>
int main()
{
char a[20];
int n=sprintf(a,"javaToint");
printf("value of n is %d",n);
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:

```c
#include<stdio.h>
#define start main
void start() {
   printf("Hello");
}
```

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:

178. Identifiers: Identifiers refer to the name of the variables.
179. Keywords: Keywords are the predefined words that are explained by the compiler.
180. Constants: Constants are the fixed values that cannot be changed during the execution of a program.
181. Operators: An operator is a symbol that performs the particular operation.
182. 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:

```c
main(int count, char *args[]){
//code to  be executed
}
```

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

```c
#include<stdio.h>
#include<conio.h>
int main()
{

 char ch;
 printf("Enter a character ");
 ch=getch(); // taking an user input without printing the value.
 printf("\nvalue of ch is %c",ch);
 printf("\nEnter a character again ");
 ch=getche(); // taking an user input and then displaying it on the screen.
  printf("\nvalue of ch is %c",ch);
 return 0;
}
```

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.

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

```
200.(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:

```
for(;;){
//code to be executed
}
```

Infinite While Loop:

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

Infinite Do-While Loop:

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

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

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

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

```
#include<stdio.h>
#include<conio.h>
main()
{
int a=10, b=20;    //declaration of variables.
clrscr();        //It clears the screen.
printf("Before swap a=%d b=%d",a,b);

a=a+b;//a=30 (10+20)
b=a-b;//b=10 (30-20)
a=a-b;//a=20 (30-10)

printf("\nAfter swap a=%d b=%d",a,b);
getch();
}
```

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

```
#include<stdio.h>
#include<conio.h>
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();
}
```

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

```
#include<stdio.h>
#include<conio.h>
void printFibonacci(int n) // function to calculate the fibonacci series of a given number.
```

```c
{
static int n1=0,n2=1,n3;    // declaration of static variables.
   if(n>0){
       n3 = n1 + n2;
       n1 = n2;
      n2 = n3;
       printf("%d ",n3);
       printFibonacci(n-1);    //calling the function recursively.
   }
}
void main(){
   int n;
   clrscr();
   printf("Enter the number of elements: ");
   scanf("%d",&n);
   printf("Fibonacci Series: ");
   printf("%d %d ",0,1);
   printFibonacci(n-2);//n-2 because 2 numbers are already printed
   getch();
}
```

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

```c
#include<stdio.h>
#include<conio.h>
void main()
{
int n,i,m=0,flag=0;    //declaration of variables.
clrscr();    //It clears the screen.
printf("Enter the number to check prime:");
scanf("%d",&n);
m=n/2;
for(i=2;i<=m;i++)
{
```

```
if(n%i==0)
{
printf("Number is not prime");
flag=1;
break;    //break keyword used to terminate from the loop.
}
}
if(flag==0)
printf("Number is prime");
getch();    //It reads a character from the keyword.
}
```

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

```
#include<stdio.h>
#include<conio.h>
main()
{
int n,r,sum=0,temp;
clrscr();
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)
{
r=n%10;
sum=(sum*10)+r;
n=n/10;
}
if(temp==sum)
printf("palindrome number ");
else
printf("not palindrome");
getch();
}
```

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

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

  for(i=1;i<=number;i++){
      fact=fact*i;
  }
  printf("Factorial of %d is: %d",number,fact);
  getch();
}
```

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

```c
#include<stdio.h>
#include<conio.h>
long factorial(int n)    // function to calculate the factorial of a given number.
{
  if (n == 0)
    return 1;
else
return(n * factorial(n-1));    //calling the function recursively.
}
void main()
{
  int number;    //declaration of variables.
  long fact;
 clrscr();
 printf("Enter a number: ");
scanf("%d", &number);
 fact = factorial(number);    //calling a function.
printf("Factorial of %d is %ld\n", number, fact);
 getch();   //It reads a character from the keyword.
}
```

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

```
#include<stdio.h>
#include<conio.h>
main()
{
int n,r,sum=0,temp;    //declaration of variables.
clrscr(); //It clears the screen.
printf("enter the number=");
scanf("%d",&n);
temp=n;
while(n>0)
{
r=n%10;
sum=sum+(r*r*r);
n=n/10;
}
if(temp==sum)
printf("armstrong  number ");
else
printf("not armstrong number");
getch();  //It reads a character from the keyword.
}
```

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

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

```c
#include<conio.h>
main()
{
int n, reverse=0, rem;    //declaration of variables.
clrscr(); // It clears the screen.
printf("Enter a number: ");
scanf("%d", &n);
while(n!=0)
{
    rem=n%10;
    reverse=reverse*10+rem;
    n/=10;
}
printf("Reversed Number: %d",reverse);
getch();  // It reads a character from the keyword.
}
```