Page No.
Date

1) What do you understand by identifiers and keywords?

→ Every C word is classified as either keyword or an identifier.

- **keyword :**
  keywords are predefined, reserved words used in programming that have special meaning to compiler. Keywords have fixed meanings and these meaning cannot be changed. They are part of syntax and they cannot be used as an identifiers. It must be written in lowercase letter and they does not contain any special character such as underscore.
- **Identifier :**
  Identifiers refer to the names of variables, functions and arrays. These are user defined names and consist of sequance of letters and digits, with letter as first character. Both uppercase and lowercase letter are permitted. The underscore character is also permitted in identifier.

2) Enlist all the primitive data type in C language along with their memory requirements, format specifiers and range.

Data type	Range	Bytes	Format
signed char	-128 to +127	1	%c
unsigned char	0 to 255	1	%c
short signed int	-32768 to +32767	2	%d
short unsigned int	0 to 65535	2	%u
Signed int	-2147483648 to +2147483647	4	%d
unsigned int	0 to 4294967295	4	%u

Teacher's Sign.: _____

long signed int	-2147483648 to +2147483647	4	%ld
long unsigned int	0 to 4294967295	4	%lu
float	-3.4e38 to +3.4e38	4	%f
double	-1.7e308 to +1.7e308	8	%lf
long double	-1.7e4932 to +1.7e4932	10	%Lf

3) Write a short Note on operators available in C language.

→ An operator is a symbol that tells the compiler to perform a certain mathematical or logical operations, based on the values provided by operators

C language is rich in built-in operators and provide the following types of operators

1) Arithmetic Operators - This operator performs mathematical operations like (+, -, *, /, %)

2) Increment and Decrement operators -
   Increment (++) increases the value by 1
   Decrement (--) decreases the value by 1

3) Assignment Operators -
   An assigment operator is used for assigning a value to a variable
   eg '=' ⇒ a = b
   '+=' ⇒ a += b ⇒ a = a+b

4) Relational operator - It checks the relationship between two operands
   eg: ==, >, <, !=, >=, <=
   True - return 1
   false - return 0

Teacher's Sign.:

5) logical operators -
An expression containing logical operator returns either 0 or 1 depending upon whether expression result true or false
eg: &&, !, ||

6) Bitwise operators -
During computation, mathematical operations converted into bit-level which makes processing faster and saves power
eg: &, |, ^, ~, <<, >>

4) Enlist Bitwise operators in C language. Explain any 2 examples.

→ Bitwise operators in C language are -
1) Bitwise AND operator (&)
2) Bitwise OR operator (|)
3) Bitwise XOR operator (^)
4) Bitwise complement operator (~)
5) Right shift operator (>>)
6) Left shift operator (<<)

- Bitwise AND (&) → It takes two operands. The output will be 1 if both bits are 1 else 0

i.e.,

X	Y	X & Y
1	1	1
1	0	0
0	1	0
0	0	0

Teacher's Sign.:

- Bitwise OR (|) → It takes two operands. The output will be 1 if any of the two bits is 1 else 0 i.e.,

X	Y	X | Y
1	0	1
0	1	1
1	1	1
0	0	0

5) Explain Various properties of an algorithm.

→ An algorithm is a set of steps to solve a particular problem. It is a set of step-by-step instructions that satisfy a certain set of properties

1) Input : An algorithm should have some inputs
2) Output : At least one output should be returned by the algorithm after the completion of the specific task based on the given inputs
3) Definiteness : Every statement of the algorithm should be unambiguous
4) Finiteness : No infinite loop should be allowed in an algorithm.
5) Effectiveness :
writing an algorithm is a prior process of actual implementation of the algorithm. So, a person should analyze the algorithm in a finite amount of time with pen and paper to judge the performance for giving the final version of the algorithm.

Teacher's Sign.

6) Explain the utility of # define and # include statements.

→ # define :-

define is a preprocessor directive that is used to define macros in a C program and it is also known as maros directive.

define directive is used to declare some constant values or ar expression with a name that can be used throughout the C program.

- Syntax

  # define C-Name value

include :-

include is a preprocessor directives that are used for file inclusion in a C-program and it is also known as a file inclusion directive

include directive is used to add the contest piece of code from a reserved header file into the code file before the compilation of C program

- Syntax

  # include < file Name >

7) Explain ternary operator with example.

→ The ternary operator is an operator used for decision making in place of longer if and else conditional statements. This is also called conditional operator.

- Syntax -

  Variable = Expression 1 ? Expression 2 : Expression 3

Example : Program to store the greatest of two no

Teacher's Sign.: _____

```
# include <stdio.h>
int main ()
{
   int m, n, max number;
   m= 100;
   n= 20;
   max number = m>=n ? m : n;
   printf (" max number %d, %d, %d , m, n, maxnumber")
    return 0;
}
```

Output: max no from 100 and 20 is 100

8) WAP to find greatest number along three entered no using ternary operators.

→

```
# include <stdio.h>
# include <conio.h>
Void main()
{
   int a, b, c, great;
   clrscr ();
   printf (" Enter 3 numbers \n");
   scanf (" %d %d %d , &a, &b, &c);

   great = (a>b && a>c) ? (a) : ((b>c) ? (b) : (c));
   printf (" The greatest number is %d ", great);
   getch();

}
```

Teacher's Sign.:

Page No.
Date

9) Explain gets() and puts() functions of c language. Comment on their parameters and return values.

→ gets():

The gets functions enables the user to enter some characters followed by the enter key. All the characters entered by the user gets stored in character array. The null character is added to the array to trdes it string.

The gets() allows to enter the space-seperated strings. It returns the string entered by user

Syntax:-

```
gets(variable-name);
```

puts():

The puts functions is very much similar to printf() function. The puts function is used to print the string on the console which is previously read by using gets() or scanf() function. The puts() function returns on integer value representing the number of characters being printed on console

Syntax:-

```
puts(variable-name);
```

10) WAP to swap to 2 numbers of a two given variables without using third variable.

Teacher's Sign.:

→
```
# include <stdio.h>
# include <conio.h>
Void main()
{
  int a, b
  clrscr();
  printf ("Enter 2 numbers");
  scanf ("%d %d", &a, &b);

  a = a+b;
  b = a-b;
  a = a-b;
  printf ("After swapping a=%d, b=%d", a, b);
  getch();
}
```

11) WAP to check numbers is palindrome or not

→
```
# include <stdio.h>
# include <conio.h>
void main()
{
  int n, temp1, temp2, rev=0;
  clrscr();
  printf ("Enter a number");
  scanf ("%d", &n);
  temp1 = n;
  while (n>0)
  {
    temp2 = n%10;
    rev = (rev*10) + temp2;
    n = n/10;
  }
```

Teacher's Sign.: _____

```
if ( temp1 == temp2)
printf (" palindrome number");
else
printf (" not palindrome");
getch();
}
```

12) WAP to generate the following pattern

A] 1
12
123
1234
12345

B] 00 01 02 03
10 11 12 13
20 21 22 23
30 31 32 33

→ A]

```
#include <stdio.h>
# include<conio.h>
void main()
{
  int i,j;
  clrscr();
  for (i=1; i<=5; i++)
  {
     for (j=1; j<=i; j++)
     {
        Printf ("%d", j);
     }
     printf ("\n")
  }
  getch();
}
```

Teacher's Sign.:

B]

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int i,j;
  clrscr();
  for(i=0; i<=3; i++)
  {
    for(j=0; j<=3; j++)
    {
      Printf("%d %d",i,j);
    }
    Printf("\n");
  }
  getch();
}
```

13) Write algorithm to calculate sum and average of first n natural numbers. Also draw its flowchart.

→

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int n, i, sum=0;
  float avg;
  clrscr();
  Printf("Enter number");
  Scanf("%d", &n);
```

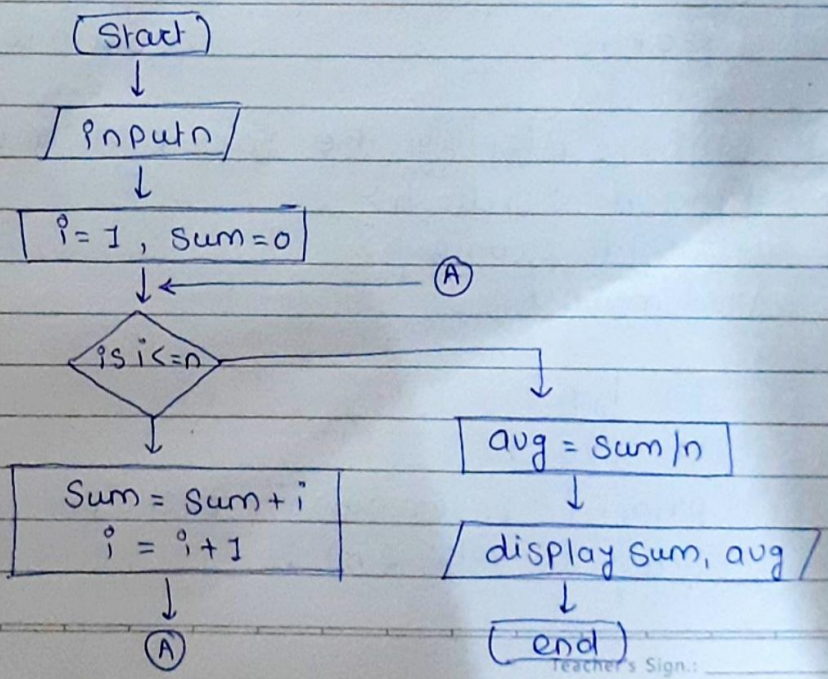
Teacher's Sign.: _____

```
for (i=0; i<=n; i++)
{
   sum = sum + i;
}
avg = sum/n;
printf("The sum is %d and the average is %f", sum, avg);
getch();
}
```

Algorithm:

Step 0 → Start
Step 1 → input n
Step 2 → i = 1, sum = 0
Step 3 → sum = sum + i
i = i + 1
} while i <= n
Step 4 → avg = sum/n
Step 5 → display sum and average
Step 6 → end

Flowchart:



Teacher's Sign.:

14) Difference bet^n Break and continue.

→

	Break statement	Continue statement
1)	The Break statement is used to exist from the loop constructs	It does not let a user to make an exit from a overall loop construct
2)	The break statement is usually used with the switch statement, & it can also use it within the while loop, do-while loop, or the for-loop.	The continue statement is not used with the switch statement, but it can be used within the while loop, do-while loop, or for-loop.
3)	When a break statement is encountered then the control is exited from the loop construct immediately	when the continue statement is encountered then the control automatically passed from the beginning of the loop statement
4)	Syntax: break;	Syntax: continue;

15) WAP to display the factor of a user entered number

→

```
#include <stdio.h>
#include <conio.h>
Void main ()
{
  int i,n;
  clrscr();
  printf("Enter number of choice");
  Scanf ("%d", &n);
```

Teacher's Sign.

```
printf ("following are the factor of given number \n");
for (i=1; i<=n; i++)
{
  if (n%. i==0)
  {
    Printf ("%d \n", i);
  }
}
  getch();
}
```

16) Explain for, while and do-while statement of C language with example

→ a) for loop :-

Syntax :

```
for (initialize ; condition ; update)
{
  // statement inside the body for loop
}
```

The initialization statement is executed only once
After initialization, the condition is checked following the body of loop if cond^n is true. After the body, the update statement is executed
If the cond^n statement is executed and comes out to be false, then the loop is terminated

Example :

```
for (i=1; i<=10; i++)
{
  Printf ("%d", i);
}
```

⇒ o/p = 1, 2, 3, 4, 5, 6, 7, 8, 9

Teacher's Sign.:

b) while loop :-

Syntax :

```
while (condition)
{
    Command;
    update;
}
```

The initialization for while loop should be done before the loop begins.

The updation statement is situated within the loop

while loop is an entry control loop.

The loop runs until the test condition is true, and the control exit the loop once the test condition become false.

Example :-

```
i = 1;
while (i<=10)
{
  printf ("%d", i);
  i++;
}
```

o/p ⇒ 1 2 3 4 5 6 7 8 9 10

c) do while loop :-

Syntax :

```
do
{
  Command;
  update;
} while (condition)
```

Teacher's Sign.: _____

The initialization for do while loop should be done before the loop begins
The update statement is situated within the loop
do while loop is an exit control loop
The loop runs until the test condition is true, and the control exits the loop once the test condition become false.
Example :-

```
i=1;
do
{
   Printf ("%d", i);
   i++;
} while (i<=10)
```

O/P :- 1 2 3 4 5 6 7 8 9 10

17) Explain the various categories of user defined functions in C with examples.

→ Types of user defined functions -

- a) function with no arguements and no return value
when we make any function with no arguments and no return value, it neither recieves any data from the calling fun^n nor returns a value. Such functions can either be used to display information, beacuse they are completely dependent on user inputs.

Input

Example :-

Teacher's Sign :

```
#include <stdio.h>
int main()
{
   greatnum();
   return 0;
}
// return type is void meaning doesn't return any value
void greatnum()
{
   int i, j;
   printf ("Enter 2 integers");
   scanf ("%d %d", &i, &j);
   if (i>j)
   printf ("The greater number is %d", i);
   else
   printf ("The greater number is: %d", j);
}
```

Output:-

Enter two integer : 12 14

The greatest no is : 14

- b) Function with no arguments and a return value

In this type of functions, arguments are passed through the calling functions to the called function but the called function has to return a value.

Example :-

```
#include <stdio.h>
int main()
{  int result;
   result = greatnum();
   printf ("The greater number is: %d", result);
   return 0;
}
```

Teacher's Sign.: _____

```
int greatnum()
{
  int i,j, greater_num;
  printf (" Enter 2 integers : ");
  scanf ("%d %d", &i, &j);
  if (i>j)
  greater_num = i;
  else
  greater_num = j;
  return greater_num;
}
```

Output:-

Enter 2 integers : 12 11

The greater number is : 12

- C) Function with arguements and no return value :

In type of function, arguements are passed through the calling function to called function but does not return value such type of functions are practically dependent on each other

Example :-

```
# include <Stdio.h>
 int main()
{
  int i,j;
  printf (" Enter two integers");
  Scanf ("%d %d", &i, &j);
  greatnum (i,j);
  return 0;
}
```

Teacher's Sign.

```
Void greatnum (int x, int y)
{
  if (x > y)
  printf (" The greater number is %d", x);
  else
  printf (" The greater number is %d", y);
}
```

Output :-

Enter two integers 45 54

The greater no is : 54

- d) Function with arguements and a return value :

This is the best type of fun^n because this makes the functions completely independant of inputs & outputs and only the logic is defined inside the function body.

Examples :-

```
# include <stdio.h>
int greatnum (int i, int j);
int main()
{
  int i, j, result;
  printf (" Enter two integers");
  scanf (" %d %d", &i, &j);
  result = greatnum (i, j);
  printf (" The greater number is %d", result);
  return 0;
}
  Void greatnum (int x, int y)
```

Teacher's Sign.: _____

```
{
  if (x>y)
  return x;
  else
  return y;
}
```

Outputs:-
Enter two integer 5 6
The greater number is 6

18) Differentiate Library functions and User defined fun^n in C & Explain with examples.

→

	User defined function	Library function
1)	A programmer creates a fun^n accordingly to the requirements of a progra-mme. Such fun^n is called as user defined fun^n	A fun^n whose prototypes are already defined in the the C library is called as the library fun^n
2)	A user defined fun^n is required to write the complete code before using the fun^n in a pro-gram.	we are not required to write a complete code to use the library function in a program
3)	The name of a user defined fun^n can be changed easily	we cannot change or modify the name of the library fun^n because the functionality of these fun^n is already defined in the compiler

Teacher's Sign.:

4)	A user defined fun^n is not compulsory to use in any C program	we need to use the library fun^n in every C-program
5)	A user defined fun^n is a part of a program	Library fun^n are part of the C header file
6)	example: Multiply(), Sum(), divide() etc	example: printf(), sqrt(), scanf(), etc

20) Explain in details nesting of functions with example.

→ A nested function is a function that is completly contained within a parent function. Any function in a program file can include a nested fun^n. The primary difference between nested function and other types of functions is that they can access and modify variable that are defined in their parent functions.

Some functions are declared inside another function. This is known as nested fun^n

Example:

```
# include <stdio.h>
main void ()
{
    auto int view(); // declare function with auto keyword
    view(); // calling function
    printf ("main\n");
    int view()
    {
      Printf ("view\n");
      return 1;
    }
    printf ("Geeks");
}
```

o/p => view
Main
Geeks

Teacher's Sign.: